Table of Contents

ZipModelSource

Introduction

Recognissimo is a cross-platform offline speech recognition plugin for Unity.

Features:

- No internet connection required
- Fast and lightweight
- Easy to use drag and drop components
- Easy to extend and modify using API
- Supports 21+ languages and dialects and more to come

Supported platforms:

- Windows (x86, x64)
- macOS (x64)
- Linux (x64)
- Android (ARMv7, ARM64, x86, API 19 or higher)
- iOS (ARM64, x64, SDK 10.0 or higher)

Supported Unity editors:

- 2019.4 and above
- Plugin for older versions can be provided upon request

Supported languages and dialects:

- Arabic
- Chinese
- English
- French
- German
- Italian
- Portuguese
- Russian
- Spanish

- Catalan, Dutch, Farsi, Filipino, Greek, Indian English, Kazakh, Swedish, Turkish, Ukrainian, Vietnamese

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

Future plans:

- New backend
- Web support
- New components and features
- More advanced examples

Known issues

- **Reference to System.IO.Compression is missing**

  Symptom:

  > error CS1069: The type name 'ZipArchive' could not be found in the namespace 'System.IO.Compression'. This type has been forwarded to assembly 'System.IO.Compression, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089' Consider adding a reference to that assembly.

  Fix: add a file *csc.rsp* (or *msc.rsp* when targeting the .NET 3.5 Equivalent scripting runtime version) to the *Assets* folder with following content:

  ```
  -r:System.IO.Compression.dll
  -r:System.IO.Compression.FileSystem.dll
  ```

Glossary

Speech recognition

The speech recognition system consists of 3 main component:

- Speech recognizer
- Model provider
- Speech source

The speech recognizer receives audio data from the speech source. It then decodes the audio data based on the language model, which contains all the data required for recognition.

> The package comes with language models for English, French, German, Spanish and Russian (*StreamingAssets/LanguageModels*).
> To add a new language model, download Vosk-compatible language models you are interested in.
>
> You can extend the current functionality and create your own model provider and speech source, read the code documentation section.

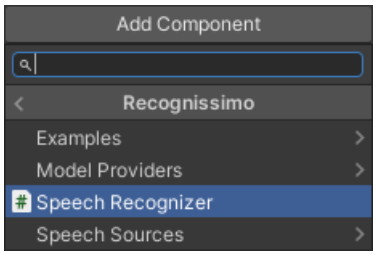
Quick start

Demo scene

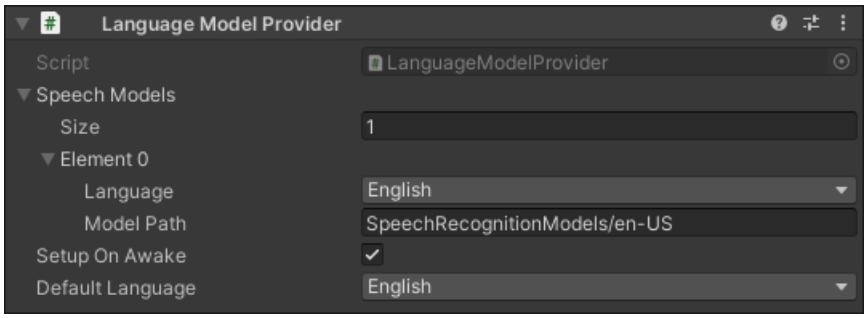
Import the package and launch the demo scene located in the *Recognissimo/Demos* folder

Speech recognition setup

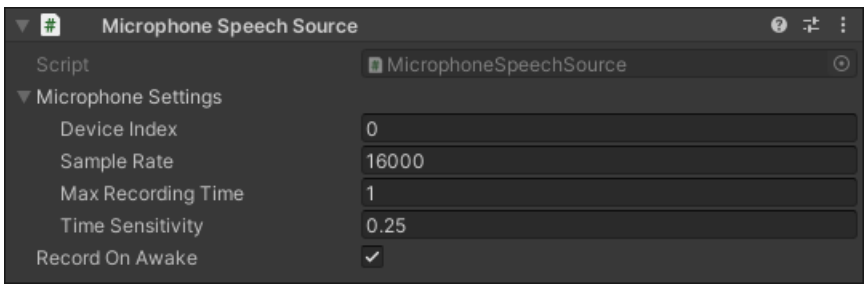
1. Setup speech recognition components
    1. Add `Speech Recognizer` component



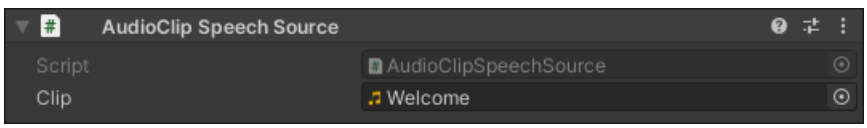
    2. Add `Language Model Provider` component. Specify path to language models relative to the *StreamingAssets* folder. Enable flag `Setup On Awake` and select desired language in `Default Language` popup menu



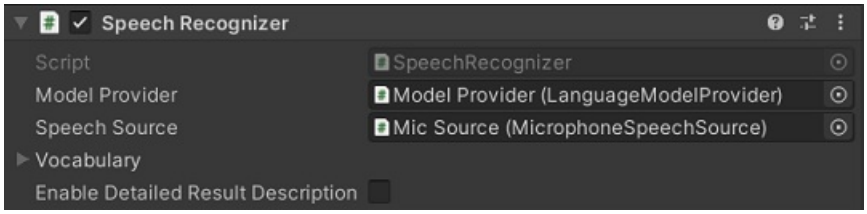
    3. Add speech source component
        - If you want to use microphone, then add `Microphone Speech Source` component. Enable flag `Record On Awake`



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



4. Connect model provider and speech source components to the speech recognizer

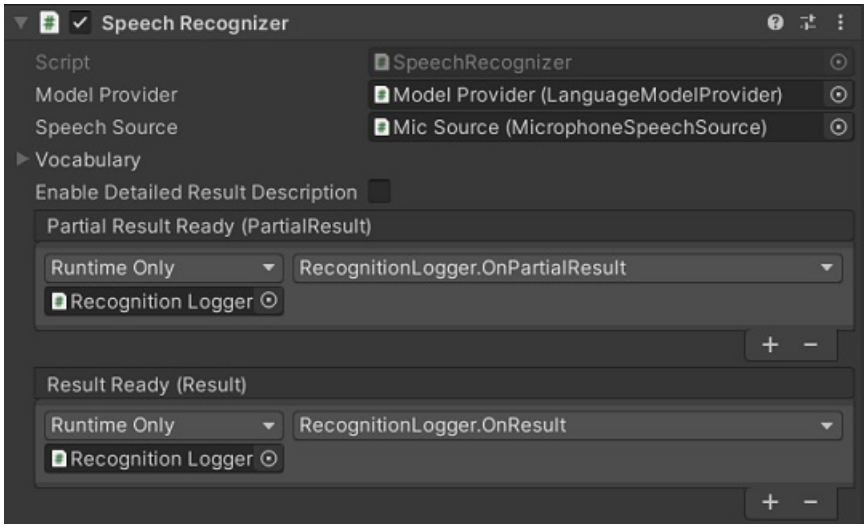


2. *(Optional)* If you want to see output:

    1. Create script called **RecognitionLogger.cs**

```
using UnityEngine;
using Recognissimo.Core; // PartialResult, Result

public class RecognitionLogger : 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. Create a new game object, add the `Recognition Logger` script to it and connect it to the `Speech Recognizer` events



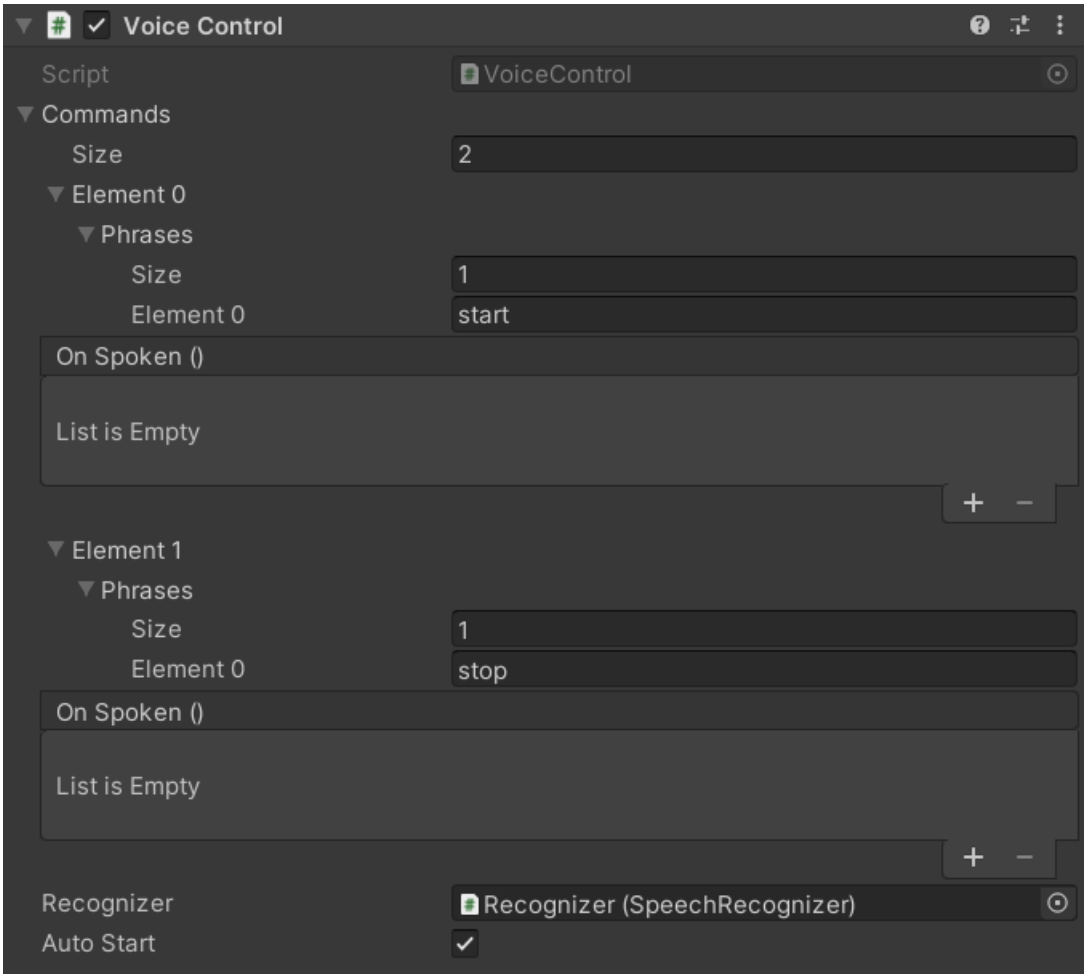
    3. Start. You should see output in console window

Voice control setup

1. Setup speech recognition components as in previous section

2. Add `Voice Control` component, assign the `Recognizer` property to the recognizer component and enable `Auto Start` flag

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

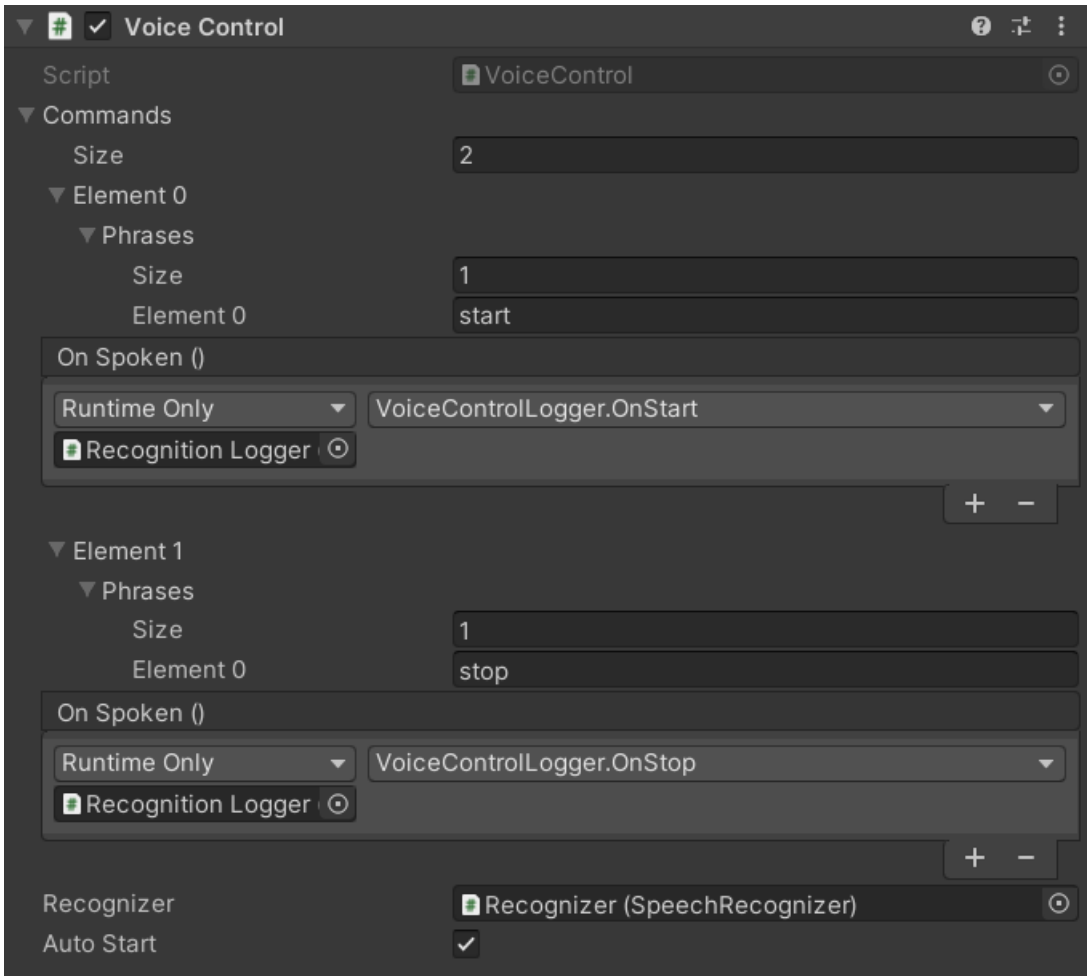


4. *(Optional)* To test voice control:

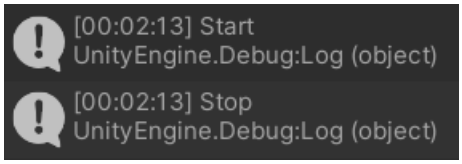
    1. Create script called VoiceControlLogger.cs

    ```
    using UnityEngine;

    public class VoiceControlLogger : MonoBehaviour
    {
        public void OnStart()
        {
            Debug.Log("Start");
        }

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

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



3. Start



Using vocabulary

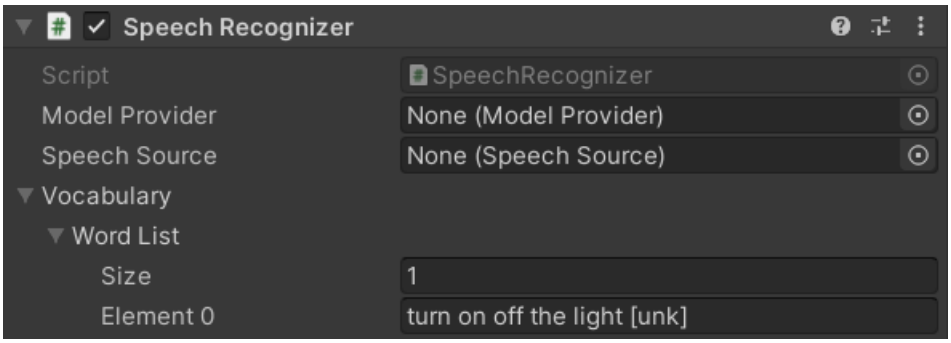
Vocabulary is a list of words available for speech recognizer. It is used to:

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

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

You can set vocabulary using:

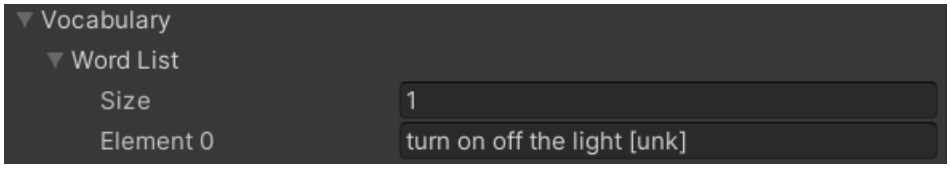
- UI (Speech Recognizer component)

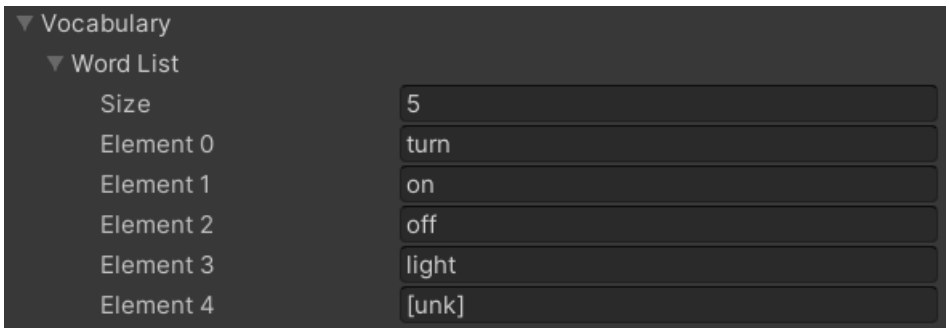


- script

```
recognizer.vocabulary.wordList = new List<string> {"turn on off light [unk]"};
```

The order of the words doesn't matter. Also you can use single line or multiple lines. The next vocabularies are the same:





Examples

Setup speech recognition using scripts

```
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 enum State
    {
        Loading,
        Ready
    };

    private State _state = State.Loading;

    // We use async operations to avoid blocking the main thread
    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}
        );
        // These operations are time-consuming, so they are performed asynchronously
        // However, synchronous versions are also available:
        //     modelProvider.Initialize();
        //     modelProvider.LoadLanguageModel(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)
    {
```

```
        _state = State.Loading;
        recognizer.StopRecognition();
        await modelProvider.LoadLanguageModelAsync(language);
        recognizer.StartRecognition();
        _state = State.Ready;
    }

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

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

Setup voice control using scripts

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

public class VoiceControlExample : MonoBehaviour
{
    // It is assumed that the recognizer is already configured
    [SerializeField]
    private SpeechRecognizer recognizer;

    [SerializeField]
    private VoiceControl voiceControl;

    // We use async operations to avoid blocking the main thread
    private async void Start()
    {
        var startCmd = new VoiceControl.VoiceCommand
        {
            phrases = new List<string> {"start"},
            onSpoken = new VoiceControl.SpokenEvent(OnStart)
        };

        var stopCmd = new VoiceControl.VoiceCommand
        {
            phrases = new List<string> {"stop"},
            onSpoken = new VoiceControl.SpokenEvent(OnStop)
        };

        voiceControl.commands = new List<VoiceControl.VoiceCommand>
        {
            startCmd, stopCmd
        };

        voiceControl.recognizer = recognizer;

        await voiceControl.SetupAsync();

        voiceControl.StartControl();
    }

    private void OnStart()
    {
        Debug.Log("Start");
    }

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

Namespace Recognissimo.Components

Classes

AudioClipSpeechSource

Sets up an AudioClip as speech source for the SpeechRecognizer

LanguageModelProvider

Model provider for different languages

MicrophoneSpeechSource

Sets up an microphone as speech source for the SpeechRecognizer

ModelProvider

Base class for all model providers

SpeechRecognizer

This is the primary Recognissimo component. It processes audio data and outputs a result based on the language model

SpeechSource

Base class for all speech sources

VoiceControl

Voice control component

Structs

LanguageModelProvider.ModelStreamingAssetsPath

Model language/path pair

LanguageModelProvider.ModelTag

Additional model info

MicrophoneSpeechSource.MicrophoneSettings

Microphone settings

SpeechRecognizer.Vocabulary

Recognizer's vocabulary

VoiceControl.VoiceCommand

Phrase/callback pair for voice control

Class AudioClipSpeechSource

Sets up an AudioClip as speech source for the SpeechRecognizer

Inheritance

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

Inherited Members

SpeechSource.SamplesReady
SpeechSource.Dried
SpeechSource.OnSamplesReady(SpeechSource.SamplesReadyEvent)
SpeechSource.OnDried()

Namespace: Recognissimo.Components

Assembly: Assembly-CSharp.dll

Syntax

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

Fields

clip

Audio clip from which the data will be taken

Declaration

```
public AudioClip clip
```

Field Value

Type	Description
UnityEngine.AudioClip	

Properties

SampleRate

Speech sampling rate. The parameter is read once at the start of recognition

Declaration

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

Property Value

Type	Description
System.Int32	

Overrides

SpeechSource.SampleRate

Methods

StartProduce()

Method called by the recognizer at the start of recognition

Declaration

```
public override void StartProduce()
```

Overrides

SpeechSource.StartProduce()

StopProduce()

Method called by the recognizer at the stop of recognition

Declaration

```
public override void StopProduce()
```

Overrides

SpeechSource.StopProduce()

Class LanguageModelProvider

Model provider for different languages

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
ModelProvider
LanguageModelProvider

Inherited Members

ModelProvider.CreateModel(String)

Namespace: Recognissimo.Components

Assembly: Assembly-CSharp.dll

Syntax

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

Fields

defaultLanguage

Language loaded by default if setupOnAwake is used

Declaration

```
[HideInInspector]
public SystemLanguage defaultLanguage
```

Field Value

Type	Description
UnityEngine.SystemLanguage	

setupOnAwake

Whether to start initialization as soon as the component awakes. If selected, defaultLanguage will be used

Declaration

```
public bool setupOnAwake
```

Field Value

Type	Description
System.Boolean	

speechModels

List of available models

Declaration

```
public List<LanguageModelProvider.ModelStreamingAssetsPath> speechModels
```

Field Value

Type	Description
System.Collections.Generic.List<LanguageModelProvider.ModelStreamingAssetsPath>	

Properties

Model

Language model instance. The parameter is read once at the start of recognition

Declaration

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

Property Value

Type	Description
Vosk.Model	

Overrides

ModelProvider.Model

Methods

Initialize()

Initialize state, load and check models. Time-consuming

Declaration

```
public void Initialize()
```

InitializeAsync()

Initialize() async variant

Declaration

```
public async Task InitializeAsync()
```

Returns

Type	Description
System.Threading.Tasks.Task	Task object

LoadLanguageModel(SystemLanguage)

Loads the model of the selected language and saves it to the Model. Time-consuming

Declaration

```
public void LoadLanguageModel(SystemLanguage language)
```

Parameters

Type	Name	Description
UnityEngine.SystemLanguage	language	Language of new model

LoadLanguageModelAsync(SystemLanguage)

LoadLanguageModel(SystemLanguage) async variant

Declaration

```
public async Task LoadLanguageModelAsync(SystemLanguage language)
```

Parameters

Type	Name	Description
UnityEngine.SystemLanguage	language	Language of new model

Returns

Type	Description
System.Threading.Tasks.Task	Task object

Struct LanguageModelProvider.ModelStreamingAssetsPath

Model language/path pair

Namespace: Recognissimo.Components

Assembly: Assembly-CSharp.dll

Syntax

```
[Serializable]
public struct ModelStreamingAssetsPath
```

Fields

language

Language of the model

Declaration

```
public SystemLanguage language
```

Field Value

Type	Description
UnityEngine.SystemLanguage	

modelPath

Path relative to StreamingAssets folder

Declaration

```
public string modelPath
```

Field Value

Type	Description
System.String	

Struct LanguageModelProvider.ModelTag

Additional model info

Namespace: Recognissimo.Components

Assembly: Assembly-CSharp.dll

Syntax

```
[Serializable]
public struct ModelTag
```

Fields

language

Language of the model

Declaration

```
public SystemLanguage language
```

Field Value

Type	Description
UnityEngine.SystemLanguage	

lastWriteTime

Time the model was installed. Field is used on Android to avoid model re-extraction from OBB

Declaration

```
public long lastWriteTime
```

Field Value

Type	Description
System.Int64	

Methods

Equals(String)

Declaration

```
public bool Equals(string json)
```

Parameters

Type	Name	Description
System.String	json	

Returns

Type	Description
System.Boolean	

Class MicrophoneSpeechSource

Sets up an microphone as speech source for the SpeechRecognizer

Inheritance

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

Inherited Members

SpeechSource.SamplesReady
SpeechSource.Dried
SpeechSource.OnSamplesReady(SpeechSource.SamplesReadyEvent)
SpeechSource.OnDried()

Namespace: Recognissimo.Components

Assembly: Assembly-CSharp.dll

Syntax

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

Fields

microphoneSettings

Microphone initialization settings. This settings will be used when recording starts

Declaration

```
public MicrophoneSpeechSource.MicrophoneSettings microphoneSettings
```

Field Value

Type	Description
MicrophoneSpeechSource.MicrophoneSettings	

recordOnAwake

Whether to start capturing as soon as the component awakes

Declaration

```
public bool recordOnAwake
```

Field Value

Type	Description
System.Boolean	

Properties

SampleRate

Speech sampling rate. The parameter is read once at the start of recognition

Declaration

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

Property Value

Type	Description
System.Int32	

Overrides

SpeechSource.SampleRate

Methods

StartMicrophone()

Start voice capture

Declaration

```
public void StartMicrophone()
```

StartProduce()

Method called by the recognizer at the start of recognition

Declaration

```
public override void StartProduce()
```

Overrides

SpeechSource.StartProduce()

StopMicrophone()

Stop voice capture

Declaration

```
public void StopMicrophone()
```

StopProduce()

Method called by the recognizer at the stop of recognition

Declaration

```
public override void StopProduce()
```

Overrides

SpeechSource.StopProduce()

Struct MicrophoneSpeechSource.MicrophoneSettings

Microphone settings

Namespace: Recognissimo.Components

Assembly: Assembly-CSharp.dll

Syntax

```
[Serializable]
public struct MicrophoneSettings
```

Fields

deviceIndex

Microphone index from UnityEngine.Microphone.devices list

Declaration

```
public int deviceIndex
```

Field Value

Type	Description
System.Int32	

maxRecordingTime

Max length of recording before overlapping (seconds). Use smaller values to reduce the delay at the start of recording. Recommended value is 1

Declaration

```
public int maxRecordingTime
```

Field Value

Type	Description
System.Int32	

sampleRate

Sampling frequency of the device (Hz). Use smaller values to reduce memory consumption. Recommended value is 16000 Hz

Declaration

```
public int sampleRate
```

Field Value

Type	Description
System.Int32	

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

Declaration

```
public float timeSensitivity
```

Field Value

Type	Description
System.Single	

Class ModelProvider

Base class for all model providers

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
ModelProvider
LanguageModelProvider

Namespace: Recognissimo.Components

Assembly: Assembly-CSharp.dll

Syntax

```
public abstract class ModelProvider : MonoBehaviour
```

Properties

Model

Language model instance. The parameter is read once at the start of recognition

Declaration

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

Property Value

Type	Description
Vosk.Model	

Methods

CreateModel(String)

Helper method to create language model from path provided. Model instantiating is time consuming. Prefer this over direct model instantiation as it handles native exceptions

Declaration

```
protected static Model CreateModel(string path)
```

Parameters

Type	Name	Description

Type	Name	Description
System.String	path	The path to the directory containing model files

Returns

Type	Description
Vosk.Model	Model instance

Class SpeechRecognizer

This is the primary Recognissimo component. It processes audio data and outputs a result based on the language model

Inheritance

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

Namespace: Recognissimo.Components

Assembly: Assembly-CSharp.dll

Syntax

```
[AddComponentMenu("Recognissimo/Speech Recognizer")]
public class SpeechRecognizer : MonoBehaviour
```

Fields

alternatives

Whether the recognition result should contain list of alternative results

Declaration

```
public int alternatives
```

Field Value

Type	Description
System.Int32	

crashed

Speech recognizer crashed

Declaration

```
public UnityEvent crashed
```

Field Value

Type	Description
UnityEngine.Events.UnityEvent	

enableDetailedResultDescription

Whether the recognition result should include details

Declaration

```
public bool enableDetailedResultDescription
```

Field Value

Type	Description
System.Boolean	

finished

Speech source dried and all samples are recognized

Declaration

```
public UnityEvent finished
```

Field Value

Type	Description
UnityEngine.Events.UnityEvent	

modelProvider

Model provider. This value is read when StartRecognition() called

Declaration

```
public ModelProvider modelProvider
```

Field Value

Type	Description
ModelProvider	

partialResultReady

New partial result ready

Declaration

```
public SpeechRecognizer.PartialResultEvent partialResultReady
```

Field Value

Type	Description
Recognissimo.Components.SpeechRecognizer.PartialResultEvent	

resultReady

New result ready

Declaration

```
public SpeechRecognizer.ResultEvent resultReady
```

Field Value

Type	Description
Recognissimo.Components.SpeechRecognizer.ResultEvent	

speechSource

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

Declaration

```
public SpeechSource speechSource
```

Field Value

Type	Description
SpeechSource	

vocabulary

Vocabulary. This value is read when StartRecognition() called

Declaration

```
public SpeechRecognizer.Vocabulary vocabulary
```

Field Value

Type	Description
SpeechRecognizer.Vocabulary	

Properties

IsRecognizing

Current recognition state

Declaration

```
public bool IsRecognizing { get; }
```

Property Value

Type	Description
System.Boolean	

Methods

StartRecognition()

Start speech recognition. Fields speechSource and modelProvider must be set by the time the method is called

Declaration

```
public void StartRecognition()
```

StopRecognition()

Stop speech recognition

Declaration

```
public void StopRecognition()
```

Struct SpeechRecognizer.Vocabulary

Recognizer's vocabulary

Namespace: Recognissimo.Components

Assembly: Assembly-CSharp.dll

Syntax

```
[Serializable]
public struct Vocabulary
```

Fields

wordList

List of 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> wordList
```

Field Value

Type	Description
System.Collections.Generic.List<System.String>	

Examples

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

```
This feature may not work with some language models
```

Class SpeechSource

Base class for all speech sources

Inheritance

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

Namespace: Recognissimo.Components

Assembly: Assembly-CSharp.dll

Syntax

```
public abstract class SpeechSource : MonoBehaviour
```

Properties

SampleRate

Speech sampling rate. The parameter is read once at the start of recognition

Declaration

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

Property Value

Type	Description
System.Int32	

Methods

OnDried()

Helper method for firing the event

Declaration

```
protected void OnDried()
```

OnSamplesReady(SpeechSource.SamplesReadyEvent)

Helper method for firing the event

Declaration

```
protected void OnSamplesReady(SpeechSource.SamplesReadyEvent e)
```

```
protected void OnSamplesReady(SpeechSource.SamplesReadyEvent e)
```

Parameters

Type	Name	Description
Recognissimo.Components.SpeechSource.SamplesReadyEvent	e	

StartProduce()

Method called by the recognizer at the start of recognition

Declaration

```
public abstract void StartProduce()
```

StopProduce()

Method called by the recognizer at the stop of recognition

Declaration

```
public abstract void StopProduce()
```

Events

Dried

Event signaling that samples have run out and will no longer be available

Declaration

```
public event EventHandler Dried
```

Event Type

Type	Description
System.EventHandler	

SamplesReady

Event signaling the arrival of new samples. The submitted samples will be added to the recognition queue

Declaration

```
public event EventHandler<SpeechSource.SamplesReadyEvent> SamplesReady
```

Event Type

Type	Description
System.EventHandler<Recognissimo.Components.SpeechSource.SamplesReadyEvent>	

Class VoiceControl

Voice control component

Inheritance

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

Namespace: Recognissimo.Components

Assembly: Assembly-CSharp.dll

Syntax

```
[AddComponentMenu("Recognissimo/Voice Control")]
public class VoiceControl : MonoBehaviour
```

Fields

autoStart

Whether to activate voice control at startup

Declaration

```
public bool autoStart
```

Field Value

Type	Description
System.Boolean	

commands

List of voice commands. The value is read when Setup() called or when script is enabled if autoStart is active

Declaration

```
public List<VoiceControl.VoiceCommand> commands
```

Field Value

Type	Description
System.Collections.Generic.List<VoiceControl.VoiceCommand>	

recognizer

Speech recognizer. The value is read when Setup() called or when script is enabled if autoStart is active

Declaration

```
public SpeechRecognizer recognizer
```

Field Value

Type	Description
SpeechRecognizer	

Methods

Setup()

Setup component. Should be called before StartControl(). Time-consuming

Declaration

```
public void Setup()
```

SetupAsync()

Setup() async variant

Declaration

```
public async Task SetupAsync()
```

Returns

Type	Description
System.Threading.Tasks.Task	Task object

StartControl()

Start voice commands processing.

Declaration

```
public void StartControl()
```

StopControl()

Stop voice commands processing

Declaration

```
public void StopControl()
```

Struct VoiceControl.VoiceCommand

Phrase/callback pair for voice control

Namespace: Recognissimo.Components

Assembly: Assembly-CSharp.dll

Syntax

```
[Serializable]
public struct VoiceCommand
```

Fields

onSpoken

UnityEvent that is triggered when phrase from the phrases is spoken.

Declaration

```
public VoiceControl.SpokenEvent onSpoken
```

Field Value

Type	Description
Recognissimo.Components.VoiceControl.SpokenEvent	

phrases

List of phrases to recognize. Case-insensitive. You can use groups "()" and alternations "|" to create options:

```
    "red|green"; // "red" and "green" will be recognized
    "turn (on|off) the light"; // "turn on the light" or "turn off the light"
```

Declaration

```
public List<string> phrases
```

Field Value

Type	Description
System.Collections.Generic.List<System.String>	

Namespace Recognissimo.Core

Classes

ModelRepository

Repository for storing language models

RecognizerWrapper

ZipModelSource

Model source for loading model from zip archive

Structs

ModelInfo

Model information

PartialResult

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

Result

Speech recognition result

Result.Alternative

Result.Word

Detailed description of decoded word

Interfaces

IModelSource

Interface for model source

IResult

Interface IModelSource

Interface for model source

Namespace: Recognissimo.Core

Assembly: Assembly-CSharp.dll

Syntax

```
public interface IModelSource
```

Properties

ModelName

Model name

Declaration

```
string ModelName { get; }
```

Property Value

Type	Description
System.String	

Methods

SaveTo(String)

Extract model files to specified folder

Declaration

```
void SaveTo(string to)
```

Parameters

Type	Name	Description
System.String	to	Extract path

Interface IResult

Namespace: Recognissimo.Core

Assembly: Assembly-CSharp.dll

Syntax

```
public interface IResult
```

Struct ModelInfo

Model information

Namespace: Recognissimo.Core

Assembly: Assembly-CSharp.dll

Syntax

```
[Serializable]
public struct ModelInfo
```

Fields

id

Model ID

Declaration

```
public string id
```

Field Value

Type	Description
System.String	

name

Model name

Declaration

```
public string name
```

Field Value

Type	Description
System.String	

path

Path to the model files

Declaration

```
public string path
```

Field Value

Type	Description
System.String	

tag

User information

Declaration

```
public string tag
```

Field Value

Type	Description
System.String	

Class ModelRepository

Repository for storing language models

Inheritance

System.Object
ModelRepository

Namespace: Recognissimo.Core

Assembly: Assembly-CSharp.dll

Syntax

```
public class ModelRepository
```

Constructors

ModelRepository(String)

Load existing or create new repository in the specified folder. Settings file is created during initialization

Declaration

```
public ModelRepository(string libraryPath)
```

Parameters

Type	Name	Description
System.String	libraryPath	Repository directory path

Methods

AddExistingModel(String, String)

Add model from local folder. Doesn't move files

Declaration

```
public ModelInfo AddExistingModel(string modelPath, string modelName)
```

Parameters

Type	Name	Description
System.String	modelPath	Model folder
System.String	modelName	Model name

Returns

Type	Description
ModelInfo	ModelInfo of installed model

Exceptions

Type	Condition
System.IO.DirectoryNotFoundException	Model folder not found
System.IO.FileNotFoundException	Model files not found in specified folder

InstallModel(IModelSource)

Install model from model source. Model files will be unpacked into repository folder

Declaration

```
public ModelInfo InstallModel(IModelSource modelSource)
```

Parameters

Type	Name	Description
IModelSource	modelSource	Model source

Returns

Type	Description
ModelInfo	ModelInfo of installed model

Models()

Get ModelInfo for existing models

Declaration

```
public IEnumerable<ModelInfo> Models()
```

Returns

Type	Description

Type	Description
System.Collections.Generic.IEnumerable<ModelInfo>	ModelInfo enumerable (empty if no models loaded)

Remove(String)

Remove model from list of models. It doesn't remove local files

Declaration

```
public void Remove(string id)
```

Parameters

Type	Name	Description
System.String	id	Model ID

SetName(String, String)

Change name of existing model

Declaration

```
public void SetName(string id, string name)
```

Parameters

Type	Name	Description
System.String	id	Model ID
System.String	name	New name

SetTag(String, String)

Change tag of existing model

Declaration

```
public void SetTag(string id, string tag)
```

Parameters

Type	Name	Description
System.String	id	Model ID

Type	Name	Description
System.String	tag	New tag

Struct PartialResult

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

Implements

IResult

Namespace: Recognissimo.Core

Assembly: Assembly-CSharp.dll

Syntax

```
[Serializable]
public struct PartialResult : IResult
```

Fields

partial

Decoded text

Declaration

```
public string partial
```

Field Value

Type	Description
System.String	

Implements

IResult

Class RecognizerWrapper

Inheritance

System.Object

RecognizerWrapper

Namespace: Recognissimo.Core

Assembly: Assembly-CSharp.dll

Syntax

```
public class RecognizerWrapper
```

Properties

EnableDetailedResultDescription

Declaration

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

Property Value

Type	Description
System.Boolean	

IsRecognizing

Declaration

```
public bool IsRecognizing { get; }
```

Property Value

Type	Description
System.Boolean	

MaxAlternatives

Declaration

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

Property Value

Type	Description
System.Int32	

SpeechModel

Declaration

```
public Model SpeechModel { get; set; }
```

Property Value

Type	Description
Vosk.Model	

Vocabulary

Declaration

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

Property Value

Type	Description
System.String	

Methods

EnqueueSamples(Single[], Int32)

Declaration

```
public void EnqueueSamples(float[] samples, int length)
```

Parameters

Type	Name	Description
System.Single[]	samples	
System.Int32	length	

GetNextResult()

Declaration

```
public IResult GetNextResult()
```

Returns

Type	Description
IResult	

Start(Int32)

Declaration

```
public void Start(int sampleRate)
```

Parameters

Type	Name	Description
System.Int32	sampleRate	

Stop()

Declaration

```
public void Stop()
```

Struct Result

Speech recognition result

Implements

IResult

Namespace: Recognissimo.Core

Assembly: Assembly-CSharp.dll

Syntax

```
[Serializable]
public struct Result : IResult
```

Fields

alternatives

List of all possible recognition results. Sorted in descending order of confidence

Declaration

```
public List<Result.Alternative> alternatives
```

Field Value

Type	Description
System.Collections.Generic.List<Result.Alternative>	

result

Detailed description of decoded text

Declaration

```
public List<Result.Word> result
```

Field Value

Type	Description
System.Collections.Generic.List<Result.Word>	

text

Decoded text

Declaration

```
public string text
```

Field Value

Type	Description
System.String	

Implements

IResult

Struct Result.Alternative

Namespace: Recognissimo.Core

Assembly: Assembly-CSharp.dll

Syntax

```
[Serializable]
public struct Alternative
```

Fields

text

Decoded text

Declaration

```
public string text
```

Field Value

Type	Description
System.String	

Struct Result.Word

Detailed description of decoded word

Namespace: Recognissimo.Core

Assembly: Assembly-CSharp.dll

Syntax

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

Fields

conf

Confidence (from zero to one)

Declaration

```
public float conf
```

Field Value

Type	Description
System.Single	

end

End time of the word (seconds)

Declaration

```
public float end
```

Field Value

Type	Description
System.Single	

start

Start time of the word (seconds)

Declaration

```
public float start
```

Field Value

Type	Description
System.Single	

word

Decoded word

Declaration

```
public string word
```

Field Value

Type	Description
System.String	

Class ZipModelSource

Model source for loading model from zip archive

Inheritance

System.Object
ZipModelSource

Implements

IModelSource

Namespace: Recognissimo.Core

Assembly: Assembly-CSharp.dll

Syntax

```
public class ZipModelSource : IModelSource
```

Constructors

ZipModelSource(String, String)

Initializes new instance from specified zip archive

Declaration

```
public ZipModelSource(string zip, string modelEntry = "/")
```

Parameters

Type	Name	Description
System.String	zip	Path to the zip archive
System.String	modelEntry	Zip archive model entry

Exceptions

Type	Condition
System.ArgumentNullException	
System.IO.FileNotFoundException	

Properties

ModelName

Model name

Declaration

```
public string ModelName { get; }
```

Property Value

Type	Description
System.String	

Methods

SaveTo(String)

Extract model files to specified folder

Declaration

```
public void SaveTo(string to)
```

Parameters

Type	Name	Description
System.String	to	Extract path

Implements

IModelSource