Speechly Client Library for Unity and C#

Overview

Speechly is an API for building voice features and voice chat moderation into games, XR, applications and web sites. Speechly Client Library for Unity and C# streams audio for analysis and provides real-time speech-to-text transcription and information extracted from the speech via the library's C# API.

Speech recognition runs by default in Speechly cloud (online). On-device (offline) capabilities are available via separate libSpeechly add-on that runs machine learning models on the device itself using Core ML and TensorFlow Lite.

Package contents

- speechly-dotnet/ contains the C# SpeechlyClient without Unity dependencies
- speechly-unity/Assets/com.speechly.speechly-unity/ folder contains the C# SpeechlyClient plus Unity-specifics like `Speechly.prefab` and `MicToSpeechly.cs` script and Unity sample projects:
  - speechly-unity/Assets/com.speechly.speechly-unity/Examples/MicToSpeechly/
  - speechly-unity/Assets/com.speechly.speechly-unity/Examples/AudioFileToSpeechly/
  - speechly-unity/Assets/com.speechly.speechly-unity/Examples/VoiceCommands/

Unity usage

Installation

- Download speechly-client.unitypackage
- Select Assets > Import Package > Custom Package... and select the downloaded file.
- Import `Speechly/` folder from the package to your Unity project
- Drop `MicToSpeechly` prefab to your scene hierarchy
- Select it and enter a valid token in the `App Id` field

Refer to https://docs.unity3d.com/Manual/upm-ui-install.html for instructions on using Unity packages.

Requirements

- Unity 2018.1 or later (tested with 2019.4.36f1 and 2021.2.12f1)
- TextMeshPro (examples only)
- XR Plug-in Management (VR example only)

Supported languages

- English
- Finnish
- Additional language support is discussed here.

Workflows

- Add the `Speechly.prefab` to your scene and select it.

- In the inspector, enter a valid `App id` acquired from Speechly dashboard
- Check `debug print` and `VAD controls listening`

`Speechly.prefab` is by default set with `Don't destroy on load` so it's available in every scene. It creates a `SpeechlyClient` singleton which you can can access with `MicToSpeechly.Instance.SpeechlyClient`. Please see SpeechlyClient API.

To display speech-to-text in a TMP text field, you can use something like this:

```
void Start()
{
  var speechlyClient = MicToSpeechly.Instance.SpeechlyClient;
  speechlyClient.OnSegmentChange += (segment) =>
  {
    Debug.Log(segment.ToString());
    TranscriptText.text = segment.ToString(
      (intent) => "",
      (words, entityType) => $"<color=#15e8b5>{words}<color=#ffffff>",
      "."
    );
  };
}
```

On-device support

Speechly Client for Unity is on-device speech recognition ready. Enabling on-device support requires add-on files (libSpeechly, speech recognition model) from Speechly.

Install libSpeechly for each target platform

- OS X (Intel): Replace the zero-length placeholder file `libSpeechlyDecoder.dylib` in `Assets/com.speechly.speechly-unity/SpeechlyOnDevice/libSpeechly/OS_X/libSpeechlyDecoder.dylib`
- Android (arm64): Replace the zero-length placeholder file `libSpeechlyDecoder.so` in `Assets/com.speechly.speechly-unity/SpeechlyOnDevice/libSpeechly/Android/libSpeechlyDecoder.so`
- Other platforms: Follow the installation instruction provided with the files.

Installing the speech recognition model

- Add the `my-custom-model.bundle` file into `speechly-unity/Assets/StreamingAssets/SpeechlyOnDevice/Models/`
- Select the `Speechly.prefab` and enter the model's filename (e.g. `my-custom-model.bundle`) as the value for Model Bundle property.

Reference

- SpeechlyClient API documentation (DocFX generated)

Samples

MicToSpeechly

Import com.speechly.speechly-unity/Examples/MicToSpeechly/ and `Speechly/` folders from speechly-client.unitypackage to run a Unity sample scene that streams data from microphone to Speechly using MicToSpeechly.cs script running on a GameObject. App-specific logic is in `UseSpeechly.cs` which registers a callback and shows speech-to-text results in the UI.

PointAndTalk

Import com.speechly.speechly-unity/Examples/PointAndTalk/ and `Speechly/` folders from `speechly-client.unitypackage` to run a Unity sample scene that showcases a point-and-talk interface: target an object and hold the mouse button to issue speech commands like "make it big and red" or "delete". Again, app-specific logic is in `UseSpeechly.cs` which registers a callback to respond to detected intents and keywords (entities).

Other Examples

- `AudioFileToSpeechly` streams a pre-recorded audio file for speech recognition.
- `HandsFreeListening` demonstrates hands-free voice input.
- `HandsFreeListeningVR` demonstrates hands-free voice input on a VR headset. Tested on Meta Quest 2.

OS X notes

To enable microphone input on OS X, set `Player Settings > Settings for PC, Mac & Linux Standalone > Other Settings > Microphone Usage Description`, to for example, "Voice input is automatically processed by Speechly.com".

Android notes

Device testing

To diagnose problems with device builds, you can do the following:

- First try running MicToSpeechlyScene.unity in the editor without errors.
- Change to Android player, set MicToSpeechlyScene.unity as the main scene and do a `build and run` to deploy and start the build to on a device.
- On the terminal start following Unity-related log lines:

```
adb logcat -s Unity:D
```

- Use the app on device. Ensure it's listening (e.g. keep `Hold to talk` button pressed) and say "ONE, TWO, THREE". Then release the button.
- You should see "ONE, TWO, THREE" displayed in the top-left corner of the screen. If not, see the terminal for errors.

Android troubleshooting

- `Exception: Could not open microphone` and green VU meter won't move. Cause: There's no implementation in place to wait for permission prompt to complete so mic permission is not given on the first run and Microphone.Start() fails. Fix: Implement platform specific permission check, or, restart app after granting the permission.
- `WebException: Error: NameResolutionFailure` and transcript won't change when button held and app is spoken to. Cause: Production builds restric access to internet. Fix: With Android target active, go Player settings and find "Internet Access" and change it to "required".
- IL2CPP build fails with `NullReferenceException` at `System.Runtime.Serialization.Json.JsonFormatWriterInterpreter.TryWritePrimitive`. Cause: System.Runtime.Serialization.dll uses reflection to access some methods. Fix: To prevent Unity managed code linker from stripping away these methods add the file `link.xml` with the following content:

```
<linker>
  <assembly fullname="System.Runtime.Serialization" preserve="all"/>
</linker>
```

Developing and contributing

We are happy to receive community contributions! For small fixes, feel free to file a pull request. For bigger changes or new features start by filing an issue.

- `./link-speechly-sources.sh` shell script will create hard links from `speechly-dotnet/Speechly/` to `speechly-unity/Assets/com.speechly.speechly-unity/` so shared .NET code remains is in sync. Please run the script after checking out the repo and before making any changes. If you can't use the script please ensure that the files are identical manually before opening a PR.
- `./build-docs.sh` generates public API documentation using DocFX from triple-slash `///` comments with C# XML documentation tags.

C# Usage with dotnet

Requirements

- A C# development environment with .NET Standard 2.0 API:
  - Microsoft .NET Core 3 or later (tested with .NET 6.0.200)

Command line usage with `dotnet`

SpeechlyClient features can be run with prerecorded audio on the command line in `speechly-dotnet/` folder:

- `dotnet run test` processes an example file, sends to Speechly cloud SLU and prints the received results in console.

- `dotnet run vad` processes an example file, sends the utterances audio to files in `temp/` folder as 16 bit raw and creates an utterance timestamp `.tsv` (tab-separated values) for each audio file processed.
- `dotnet run vad myaudiofiles/*.raw` processes a set of files with VAD.