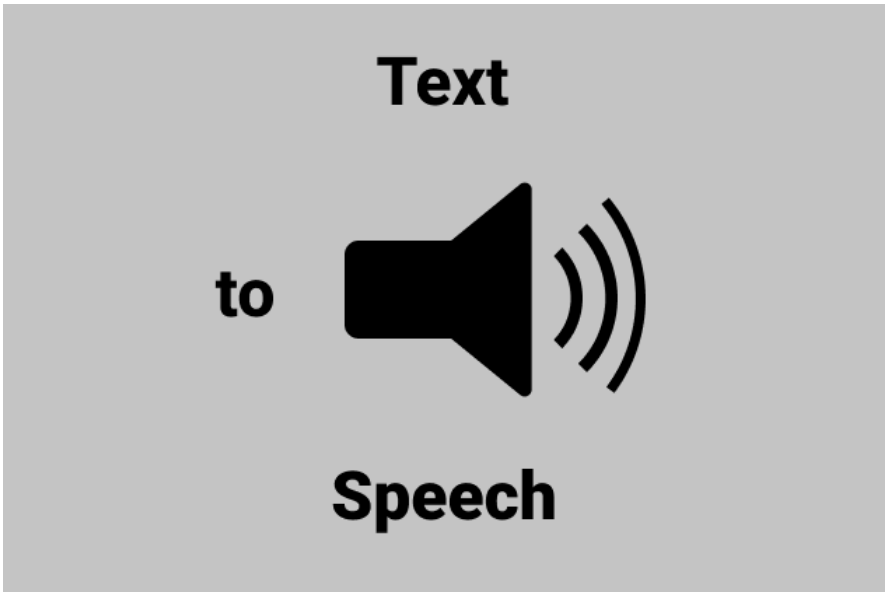


Native Text to Speech Unity plugin

Version 1.0

Created by Sergey Okhotnikov

https://okhotnikov.net/

Native Text to Speech Unity plugin

1. Introduction and Overview

Native Text to Speech plugin enables you, the Unity developer, to get benefits from native text to speech functions of iOS and Android platforms.

This solution uses speech recognition classes from iOS (AVSpeechSynthesizer) and Android (android.speech.tts.TextToSpeech).

Speech generation is offline for latest iOS and Android versions.

If you have any questions, feedback or having issues, please contact me directly at segey@okhotnikov.net. I will respond to you as quickly as possible.

2. Package content

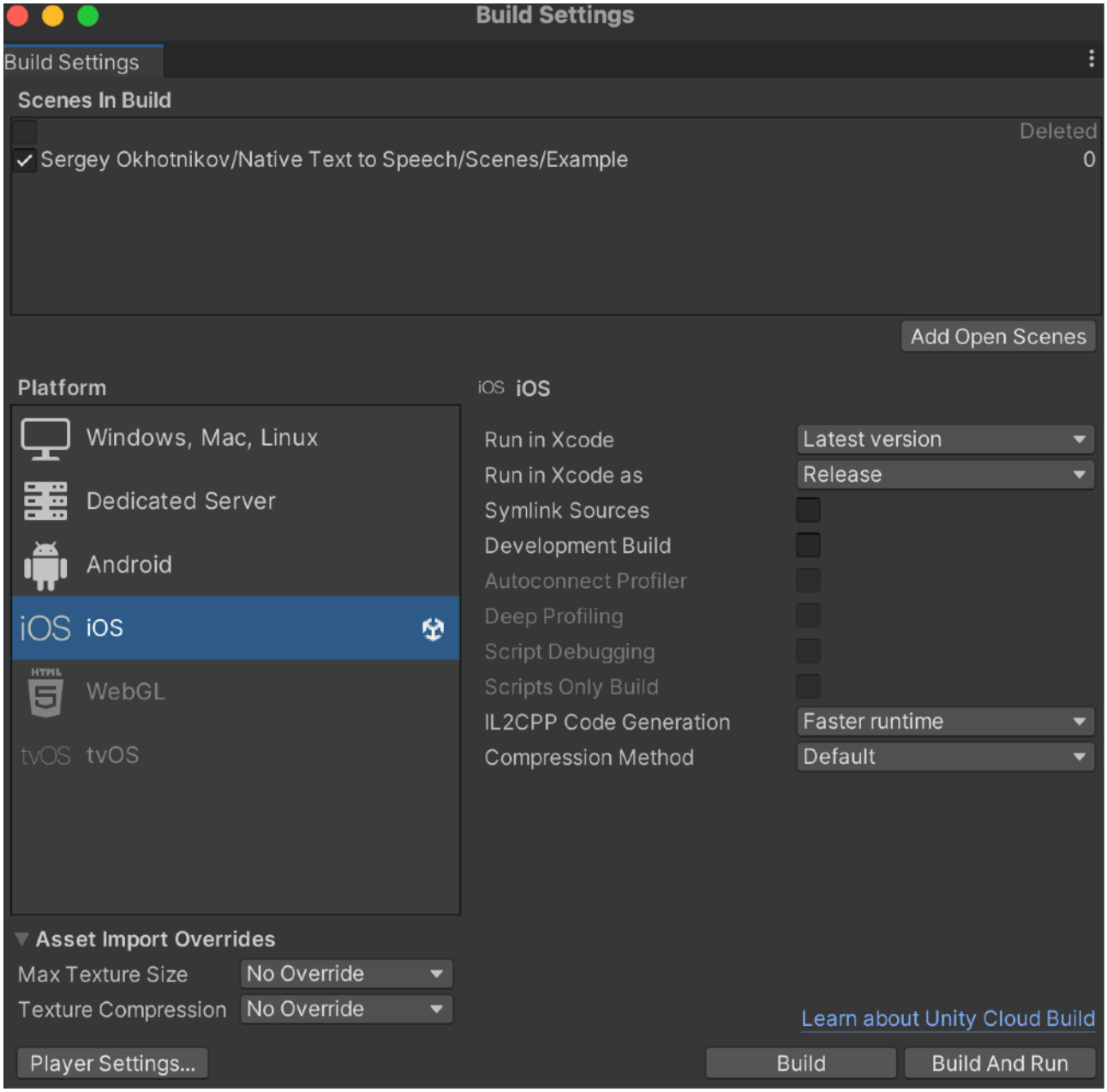
The package contains c# class that communicates with iOS and Android libraries as well as demo scene.

Module	Description
Native Text to Speech/Scenes/ Example.unity	Demo scene. You should include it into your iOS or Android build to quick start project demo.
Native Text to Speech/Scripts/ TextToSpeechExample.cs	This demo script provides interaction between demo scene and Text to Speech plugin. You should use this script as an example for you own plugin integration.
Native Text to Speech/Plugins/iOS/ TextToSpeechIOS.framework	iOS library. You shouldn't edit files in this folder.
Native Text to Speech/Plugins/ Android	Android libraries folder. That folder contains AndroidManifest and java library archive.
Native Text to Speech/Plugins/ TextToSpeech.cs	This file contains TextToSpeech class that works as a bridge between native and Unity code.

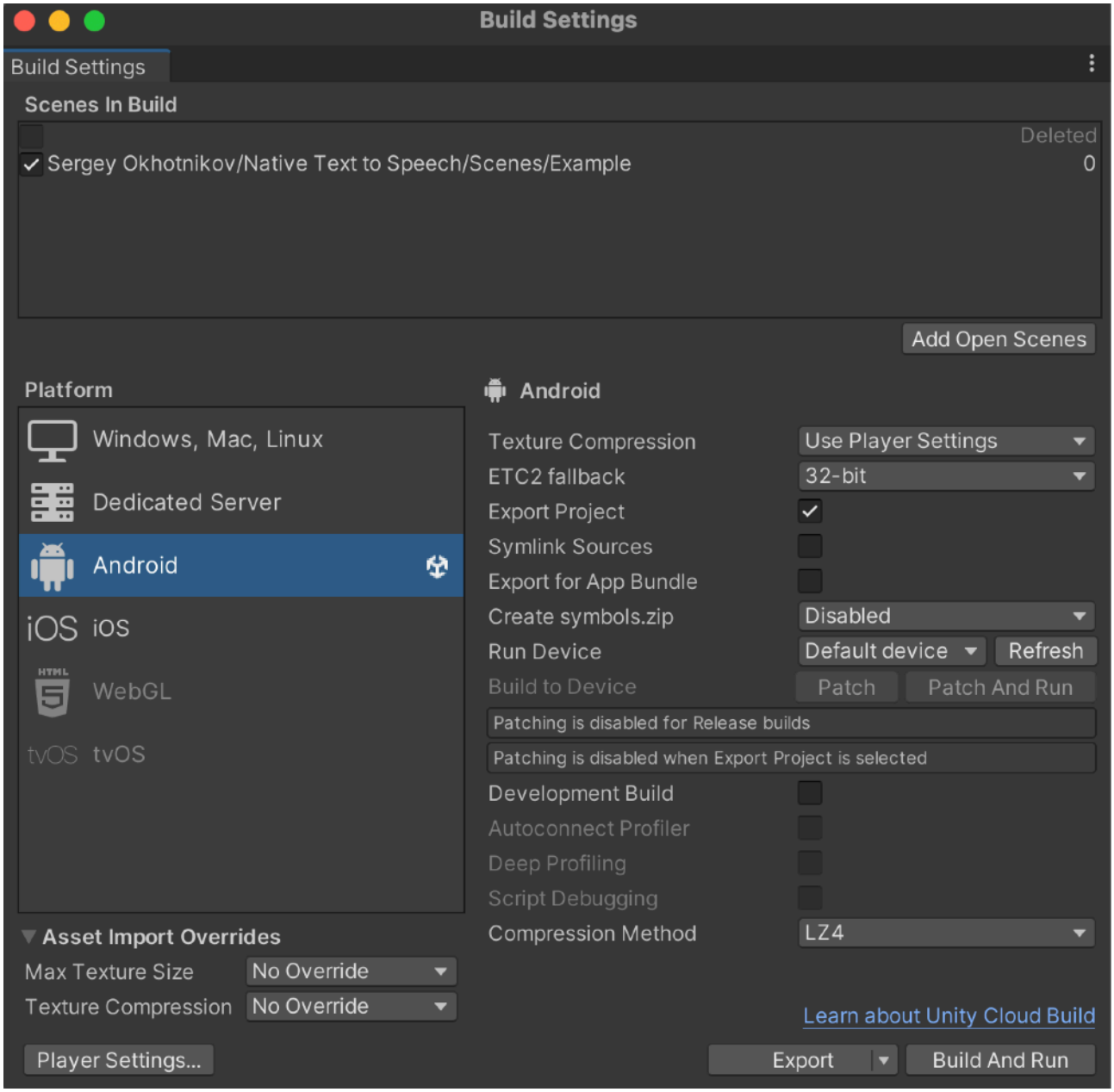
3. Quick start demo

Native Text to Speech plugin utilizes iOS and Android classes. That's why running project in Unity editor does nothing. You should create iOS or Android build to run test scene.

As a first step you should include Example.unity into your build.

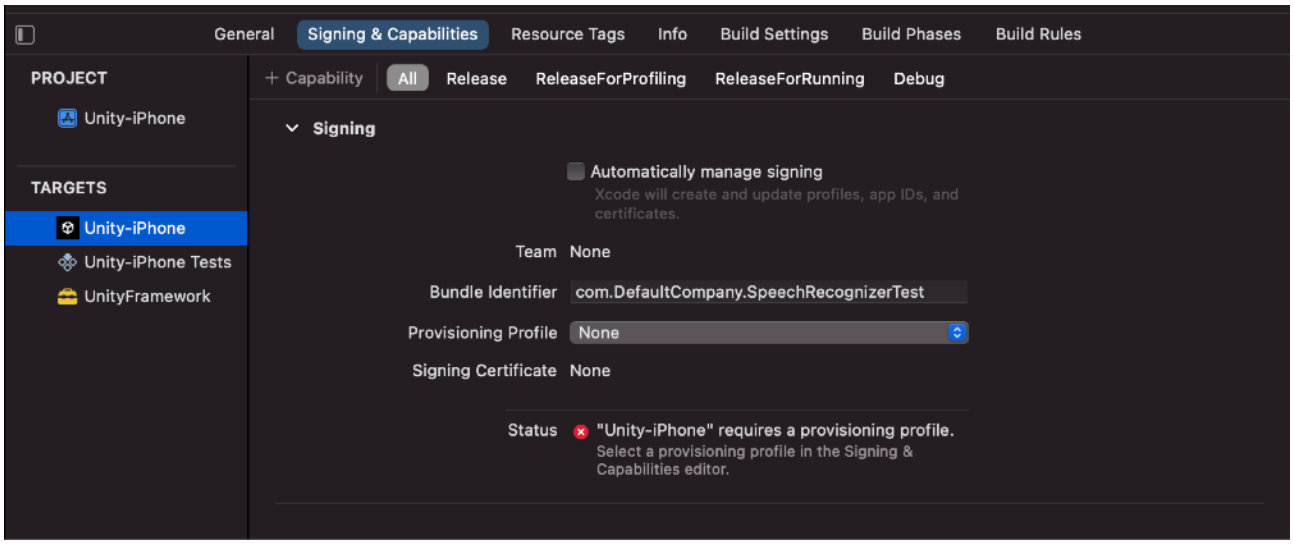


Step 1 iOS: Build a project with ExampleScene.



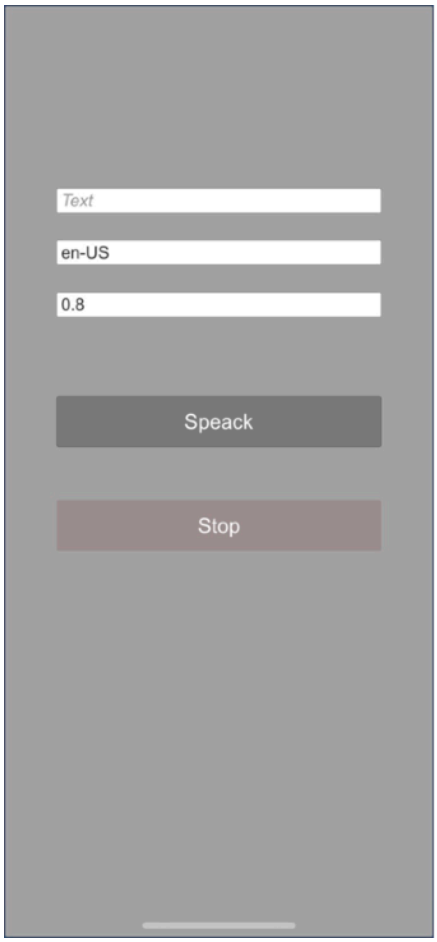
Step 1 Android: Build a project with ExampleScene.

For iOS project additional step required. You should sign your target in Xcode. Select Unity-iPhone target then Signing & Capabilities tab. Choose your project team Provisioning Profile.



Step 2 iOS: Sign the target

Now your could connect your phone and run the project.



Final step: Run the project

You could set different text, language and speechRate (speed) for every text to speech request.

4. Integration Guide

The integration process of Native Speech Recognition plugin is easy:

Step 1: Define callbacks for receiving finish event and error messages

You will need 2 methods for handling callbacks from iOS and Android libraries. One for handling event when speech is finished.

```
private void OnFinish()
{
    if (threadSafe)
    {
        _finishReceived = true;
    }
    else
    {
        TTSFinished();
    }
}
```

Second for handling error messages:

```
private void OnError(string msg)
{
    ErrorText.text = msg;
    if (msg != noError)
    {
        RecognitionNotStarted();
    }
}
```

You should implement custom logic for both methods.

Step 2: Create instance of TextoToSpeech class

Your only need one instance of TextoToSpeech at a time so it follows the singleton pattern and has private constructor. To create an instance you need to call static Create method.

```
_textToSpeech =  TextToSpeech.Create(OnFinish,OnError);
```

Parameters:

1. Action OnFinish - callback for receiving finish event
2. Action<string> error - callback for receiving platform error messages.

If everything went well you will receive noError message into your error handling callback.

Step 3: Start Text to Speech

All you need is to call TextToSpeech object's method Speak.

```
_textToSpeech.Speak("Hello world", "en-US", 0.8f);
```

Warning: iOS and Android systems have different understanding of speech speed. Normal speech speed for iOS is 0.5f. Normal speech speed for Android is 0.8f-1.0f

After speech will be played your onFinish method will be called.

Step 4: Stop playing a speech

Just call TextToSpeech object's method Stop.

```
_textToSpeech.Stop();
```

You could reuse TextToSpeech object and call Speak again when it's required.

Step 5: Stop speech recognition

Don't forget to dispose TextToSpeech object by setting variable to null.

```
_textToSpeech = null;
```

6. Handling errors

Both iOS and Android libraries almost never produce an error. If a language not found a speech will be played with less quality. Some information about that could be found in application logs.

1. **noError** - Do nothing. Everything is ok.
3. **nullInput** - Text or Language is null
4. **initializationFailed, errorInProgress, errorStartingSpeech, errorStoppingSpeech** - some error occurred on corresponding process stage (Android only)

Warning: Android libraries don’t work on emulator.

7. Supported languages iOS

1. ar-SA
2. cs-CZ
3. da-DK
4. de-DE
5. el-GR
6. en-AU
7. en-GB
8. en-IE
9. en-IN
10. en-US
11. en-ZA
12. es-ES
13. es-MX
14. fi-FI
15. fr-CA
16. fr-FR
17. he-IL
18. hi-IN
19. hu-HU
20. id-ID
21. it-IT
22. ja-JP
23. ko-KR
24. nl-NL
25. no-NO
26. pl-PL
27. pt-BR
28. pt-PT
29. ro-RO
30. ru-RU
31. sk-SK
32. sv-SE
33. th-TH
34. tr-TR
35. zh-CN
36. zh-HK
37. zh-TW

8. Supported languages Android

1. ko_KR
2. mr_IN
3. ru_RU
4. zh_TW
5. hu_HU
6. th_TH
7. ur_PK
8. nb_NO
9. da_DK
10. tr_TR
11. et_EE
12. bs
13. sw
14. pt_PT
15. vi_VN
16. en_US
17. sv_SE
18. ar
19. su_ID
20. bn_BD
21. gu_IN
22. kn_IN
23. el_GR
24. hi_IN
25. fi_FI
26. bn_IN
27. km_KH
28. fr_FR
29. uk_UA
30. pa_IN
31. en_AU
32. nl_NL
33. fr_CA
34. lv_LV
35. sr
36. pt_BR
37. de_DE
38. ml_IN
39. si_LK

40. ku
41. cs_CZ
42. pl_PL
43. sk_SK
44. it_IT
45. fil_PH
46. ne_NP
47. ms_MY
48. hr
49. en_NG
50. nl_BE
51. zh_CN
52. es_ES
53. cy
54. ja_JP
55. ta_IN
56. bg_BG
57. sq
58. yue_HK
59. en_IN
60. es_US
61. la
62. jv_ID
63. in_ID
64. te_IN
65. ro_RO
66. ca
67. en_GB