Klattersynth TTS for Unity

Small fully embedded text-to-speech engine from Strobotnik Ltd.

Table of Contents

Introduction

Klattersynth TTS is the first asset of its kind available for the Unity cross-platform engine, meaning it’s the first small and fully embedded speech synthesizer which sounds the same on every platform, and available for practically all platforms targeted by Unity. So it does not need the underlying platform (OS or browser) to offer speech features, and it does not need a network connection for external generation.

Also there is no need to pre-generate the samples before creating a build of your app or game. Klattersynth will still dynamically talk what you ask it for. It can generate and play streamed speech in real-time, or alternatively speech clips can be pre-generated with optional automatic cache of most recently used clips. By default an English Text-To-Speech algorithm is used to transform text to phonemes, which are used as an intermediate step in speech synthesis. Alternatively you can also enter text as phonemes directly, skipping the rules for English TTS conversion.

Klattersynth TTS is contained in a ~100 KB cross-platform DLL & compresses to less than 30 KB!

Minimum supported Unity version is 5.0.0.

HINT – Also check out this asset’s official web page: www.strobotnik.com/unity/klattersynth/

Example Scenes

Klattersynth/Examples folder contains the following three example scenes.

KlattersynthTTS_Example_Pangrams

A scripted dialogue between a few characters. Demonstrates a few different voice settings. Also exposes a weakness: since the text-to-speech part is just a simplified algorithm, sometimes words are not pronounced correctly. As hinted in the example, such cases can be fixed manually by entering phonemes for a word inside brackets. The demo also shows how to synchronize loudness of speech audio to position or scale of an object's transform.

KlattersynthTTS_Example_LoremIpsumRobots

Some robots positioned in a simple 3D scene, constantly talking small "lorem ipsum" excerpts. The intention is to show that the synthesized speech clips work properly through the Unity Audio Source component, so also features such as 3D spatialization and reverb zones work. When using spatialized audio, you must manually add an Audio Source in addition to a Speech component, and then change the Spatial Blend value. The demo also shows how to synchronize loudness of speech audio to position of an object's transform (moving jaws of robots).

Note that Unity has limited audio capabilities in WebGL builds, so reverb zones or other audio filtering effects cannot be demonstrated there.

KlattersynthTTS_Example_TextEntry

Free text entry to test own text input. In addition to the text field for entering text to speak, there are controls to pick example voice type, voice base frequency and the voicing source. The scene is also an example how to connect other systems, such as new Unity UI, to a speech component, through a simple helper component (`KlattersynthTTS_Example_TextEntry_Controller.cs`).

Configuring Speech Component in Unity’s Inspector

To add a Speech component to a game object, press *Add Component* button in Inspector view of the game object, and pick `Scripts > Strobotnik.Klattersynth > Speech`. If you want to set Spatial Blend value for 3D audio, you also have to manually add an Audio Source component and adjust it. Otherwise the Speech component will automatically add an Audio Source if needed.

Following table describes fields in the *Speech* component:

Use Streaming Mode	Determines if speech is synthesized “on the fly”, or if a whole speech clip is always generated (and cached) before playing it. (Forced to disabled on WebGL as there’s no support for streamed audio.)
Max Auto Cached Clips	Maximum number of generated and recently used speech clips to retain. This field is only used when streaming mode is not used.
Voice Base Frequency	Base frequency of the voice in Hz. Has no effect when Whisper voicing source is used.
Voicing Source	Source of the voicing waveform. Synthesized natural wave or a noise-based wave for whispering sound.
Ms Per Speech Frame	Speech synthesis is divided into tiny “frames”, with the amount of frames varying for each phoneme. This value determines how many milliseconds to use per frame, meaning effectively how slow the speech will be. Examples: 5 is very fast, 10 is roughly normal, 15 is slow.
Flutter	Amount of flutter to apply to voice. Typically voice has a bit of mostly unnoticeable flutter (variance).
Flutter Speed	Speed of the flutter. In addition to tweaking typical flutter, can be used to make an “unsteady” voice by making this field very small and increasing the amount of flutter.

Speech Component API

It’s recommended you add `using Strobotnik.Klattersynth;` to your `C#` scripts when writing custom code to interact with the speech. Following list describes the public methods available in the *Speech* component:

```
bool isTalking()
```
Returns true when speech is being played.

```
float getCurrentLoudness()
```
Returns the current loudness level of speech, calculated from a short range of samples.

```
string getPhonemes()
```
Returns phonemes translated and used for the last call to `speak()` or `pregenerate()`.

```
void speak(SpeechClip pregenSpeech)
```
Plays a pre-generated speech clip.

```
void speak(string text, bool bracketsAsPhonemes = false)
```
Speaks given text. If `bracketsAsPhonemes` is true, text inside brackets [...] will be considered phonemes "as is".

```
void speak(int voiceBaseFrequency, SpeechSynth.VoicingSource voicingSource,
           string text, bool bracketsAsPhonemes = false)
```
Speaks given text with given voice base frequency and voicingSource. If `bracketsAsPhonemes` is true, text inside brackets [...] will be considered phonemes "as is". In non-streaming mode this will also automatically use cached clips or pregenerate and add to cache as needed.

```
void speak(StringBuilder text, bool bracketsAsPhonemes = false)
```
Speaks given text. If `bracketsAsPhonemes` is true, text inside brackets [...] will be considered phonemes "as is". In non-streaming mode this will also automatically use cached clips or pregenerate and add to cache as needed.

```
void speak(int voiceBaseFrequency, SpeechSynth.VoicingSource voicingSource,
           StringBuilder text, bool bracketsAsPhonemes = false)
```
Speaks given text with given voice base frequency and voicingSource. If `bracketsAsPhonemes` is true, text inside brackets [...] will be considered phonemes "as is". In non-streaming mode this will also automatically use cached clips or pregenerate and add to cache as needed.

```
void pregenerate(string text, bool bracketsAsPhonemes = false)
```
Pregenerates given text and caches the result, so that cached clip is played when asked to `speak()` the same text. If `bracketsAsPhonemes` is true, text inside brackets [...] will be considered phonemes "as is". Mainly meant to be used in non-streaming mode.

```
void pregenerate(out SpeechClip speechClip, string text,
                 bool bracketsAsPhonemes = false, bool addToCache = false)
```
Pregenerates given text to a given speech clip reference. If `bracketsAsPhonemes` is true, text inside brackets [...] will be considered phonemes "as is". Mainly meant to be used in non-streaming mode.

```
void pregenerate(out SpeechClip speechClip, StringBuilder text,
                 bool bracketsAsPhonemes = false, bool addToCache = false)
```
Pregenerates given text to a given speech clip reference. If `bracketsAsPhonemes` is true, text inside brackets [...] will be considered phonemes "as is". Mainly meant to be used in non-streaming mode.

```
void pregenerate(out SpeechClip speechClip, int voiceBaseFrequency,
                 SpeechSynth.VoicingSource voicingSource, StringBuilder text,
                 bool bracketsAsPhonemes = false, bool addToCache = false)
```
Pregenerates given text, with given voice base frequency and voicingSource, to a given speech clip reference. If `bracketsAsPhonemes` is true, text inside brackets [...] will be considered phonemes "as is". Mainly meant to be used in non-streaming mode.

```
void schedule(SpeechClip speechClip)
```
Schedules a pre-generated speech clip to be played when current one ends.

```
void schedule(string text, bool bracketsAsPhonemes = false)
```
Schedules given text to speak when current one ends. If `bracketsAsPhonemes` is true, text inside brackets [...] will be considered phonemes "as is".

```
void schedule(StringBuilder text, bool bracketsAsPhonemes = false)
```
Schedules given text to speak when current one ends. If `bracketsAsPhonemes` is true, text inside brackets [...] will be considered phonemes "as is".

```
void schedule(int voiceBaseFrequency, SpeechSynth.VoicingSource voicingSource,
              string text, bool bracketsAsPhonemes = false)
```
Schedules given text to speak when current one ends, with given voice base frequency and voicingSource. If `bracketsAsPhonemes` is true, text inside brackets [...] will be considered phonemes "as is". In non-streaming mode this will also automatically use cached clips or pregenerate and add to cache as needed.

```
void schedule(int voiceBaseFrequency, SpeechSynth.VoicingSource voicingSource,
              StringBuilder text, bool bracketsAsPhonemes = false)
```
Schedules given text to speak when current one ends, with given voice base frequency and voicingSource. If `bracketsAsPhonemes` is true, text inside brackets [...] will be considered phonemes "as is". In non-streaming mode this will also automatically use cached clips or pregenerate and add to cache as needed.

```
void stop(bool allScheduled = false)
```
Stops speaking. If `allScheduled` is false, then only current one is stopped and next potentially scheduled will start. If `allScheduled` is true, speaking is stopped and all scheduled talk is erased.

List of Recognized Phonemes

Phonemes used by the speech synth are represented with ASCII characters, with some of them using 2 characters. These phonemes can be used inside […] brackets when asking the Speech component to speak using `bracketsAsPhonemes=true` flag. For example: `[z@Un]` (”zone”).

This table lists the phonemes with an example English word where the phonetic sound appears.

Vowels:		**Diphthongs:**		**Consonants:**	
i	bead – [bid]	eI	day – [deI]	N	sing – [sIN]
I	bid – [bId]	@U	go – [g@U]	T	thin – [TIn]
e	bed – [bed]	aI	eye – [aI]	D	then – [Den]
&	bad – [b&d]	aU	cow – [kaU]	S	shed – [Sed]
A	bard – [bAd]	oI	boy – [boI]	Z	beige – [beIZ]
0	(zero) cod – [k0d]	I@	beer – [bI@]	tS	etch – [ettS]
o	(small o) oval – [ov@l]	e@	bare – [be@]	dZ	edge – [eddZ]
O	(capital O) cord – [kOrd]	U@	tour – [tU@]	x	ship – [xIp]
U	good – [gUd]	O@	north – [nO@T]		
u	food – [fud]	oU	goat – [goUt]		
V	bud – [bVd]				
3	(three) bird – [b3d]	**Additional phonemes with no specific info:**			
@	about – [@'baUt]	p t k b d g m n f v s z r l w h j			

Additionally following characters are special cases:

'	(plain apostrophe) – apply primary stress
,	(comma) – apply secondary stress
+	(plus) – apply tertiary stress
	(space character) – Simple short quiet pause.

Acknowledgments

Internals of this asset are based on CC0-licensed (or public domain) speech synth found in the *SoLoud* audio engine written by Jari Komppa, which itself was based on *rsynth* by the late Nick Ing-Simmons (et al).

Questions and Answers

Here are answers to some questions you might have.

Why does Klattersynth TTS pronounce some words incorrectly?

English language does not have direct mapping from written text to phonemes. The text-to-speech part is just a simplified algorithm which knows the most common cases. Fully correct pronunciation would require including a huge phonetic dictionary of English words, increasing the small size of this asset to several megabytes.

Feedback, Feature Suggestions and Bug Reports

Send by email: contact@strobotnik.com. Write “Klattersynth TTS” in the subject.