ReadSpeaker
pioneering voice technology

ReadSpeaker Unity Gaming Plugin

Version 2.1, March. 20, 2023

Introduction

This is an introduction to using ReadSpeaker TTS in Unity. Below you will find a guide to get it installed and running. Further reading is presented in the pages listed below:
SSML
Optimization
Android

Installation

To install ReadSpeaker TTS, import the unity package to your project, once the import is finished, restart the Unity Editor.

Installing Voice Engines

To install more voice engines, import the unity package containing the voice files to your project. Restart the Unity Editor in order to start using the newly installed voice engine.

Getting Started

This section requires you to have ReadSpeaker TTS installed in your project, along with at least one voice engine.
To start using ReadSpeaker TTS, start by creating any GameObject or select an already existing GameObject and add a TTSSpeaker component to it. Once a TTSSpeaker has been added to a GameObject you can inspect the component in the inspector.
To set up the TTSSpeaker for use, begin by giving it a reference to an AudioSource component. If you are giving a voice to an in-game character, it is advised to attach the AudioSource and the TTSSpeaker to the character's root GameObject.
To continue, add an AudioSource component to the same GameObject to which you attached the TTSSpeaker component to earlier and reference it in the speakers' Audio Source field. Next up we will define the speech characteristics of our newly created speaker.
You can select whether you would like to use a preset voice or define the characteristics explicitly for this speaker. For now "Use Preset" can be set to false. The next step is to choose a voice engine from the dropdown menu. Select any voice engine installed
in your project. You will be presented with some information about the selected voice engine such as its gender and language. Below that you will find a number of fields that allows you to further customize the voice by adjusting its volume, pitch, speed, etc.
For more information about the adjustable values, see TTSSpeechCharacteristics. You can preview the effects immediately in the Editor by pressing the "Preview" button below. The speaker is now ready to be used during runtime. Below is an example of how to use the TTSSpeaker component to perform real-time TTS.

```
using UnityEngine;
using ReadSpeaker;

public class TTSTest : MonoBehaviour{
private TTSSpeaker speaker;

public void Start(){
    TTS.Init();
    speaker = GetComponent<TTSSpeaker>();
}

public void Update(){
    if(Input.GetKeyDown(KeyCode.Space)){
        TTS.SayAsync("Spacebar was pressed!", speaker);
    }
  }
}
```

By attaching this script to the same GameObject which already holds the TTSSpeaker as well as the AudioSource component we should be able to start the game and listen to the voice speak as we press space.

Note regarding Say/SayAsync

Note that here we used TTS.SayAsync as opposed to TTS.Say which
allowed us to perform the computationally heavy operation of synthesis on a background thread. The tradeoff is uncertainty as to when the synthesis completes. Depending on your use case, you might want to use either of these variants. If you rely on a steady
framerate, TTS.SayAsync is preferred as only a small fraction of time is spent in the main thread. If you want to rely on the speech being played the next frame after the call, TTS.Say should be used.

SSML

ReadSpeaker TTS supports Speech Synthesis Markup Language (SSML). SSML and tags are automatically padded to the input text when supplying the TextType.SSML argument to TTS.Say/TTS.SayAsync.
To synthesize using SSML, pass the TextType.SSML argument to the TTS.Say/TTS.SayAsync function like so:

```
string ssmlText = "<voice name=\"james\"> My name is james </voice> <voice
name=\"ashley\"> and my name is Ashley </voice>"
TTS.Say(ssmlText, TextType.SSML);
```

For more information, please refer to the English-ReadSpeaker-SSML-Manual.pdf manual, and the W3C SSML Standard.

Optimization

This section details usage of the TTS can be optimized to increase performance or reduce memory usage.

Say vs. SayAsync

Using TTS.SayAsync offloads the computationally heavy operation of synthesizing to a background thread. Using TTS.Say, the synthesis takes place on the main thread. Depending on your use case, you might want to use either of these variants. If you rely on a steady framerate, TTS.SayAsync is preferred as only a small fraction of time is spent in the main thread. If you want to rely on the speech being played the next frame after the call, TTS.Say should be used.

Loading/Unloading voice engines

In the case that the engine is not currently loaded into memory upon synthesis, it will load into memory automatically. To avoid occupying memory the voice engine is then immediately unloaded once it has
been determined that it is no longer used. The loading operation can be computationally expensive, as such there may be cases where it is more beneficial to keep the engine loaded in memory for a longer period of time. To manually control when a voice engine gets loaded and unloaded from memory, use TTSEngine.Load() and TTSEngine.Unload().

Manual TTS conversion

There might be cases where the workload of performing the synthesis should be separated from playing the audio. For these purposes, use the TTSConverter class. This class is used by TTS.Say and TTS.SayAsync in the backend to perform synthesis. An example is shown below:

```
using System.Collections.Generic;
using UnityEngine;
using ReadSpeaker;

public class TTSFactory : MonoBehaviour{
    public void Start(){
        TTS.Init();
        List<float[]> results = new List<float[]>();
        TTSConverter converter = new TTSConverter();
        TTSEngine engine = TTS.GetEngine("ashley","d16");
        converter.Engine = engine;
        converter.Volume = 250;
        converter.Pitch = 50;
        converter.Speed = 125;
        converter.Pause = 0;
        converter.CommaPause = 0;
        converter.EmphasisFactor = 0;
        converter.TextType = TextType.Normal;
        converter.IsAsync = false;
        for(int i = 0; i < 100; i++){
            converter.Pitch = 50 + (i*2);
            converter.Text = i.ToString();
            converter.ConvertToBuffer();
            results.Add(converter.GetAudioData());
        }
    }
}
```

The above code would result in a list of 100 audio data arrays each containing the audio data that was produced by reading their index of the list.

Multithreading

When converting on threads other than the main thread make sure to use a separate TTSConverter for each thread. Also consider using the thread safe implementations of ConvertToBuffer and ConvertToFile: ReadSpeaker.TTSConverter.ConvertToBufferThreadProc and ReadSpeaker.TTSConverter.ConvertToFileThreadProc.
This will avoid race conditions by locking voice engines to perform one conversion at a time. Different voice engines can convert in parallel. Make sure to set TTSConverter.IsAsync to true when running on a separate thread. For example:

```
using System.Threading;
using UnityEngine;
using ReadSpeaker;

public class TTSFactory : MonoBehaviour{
    public void Start(){
        TTS.Init();
        TTSEngine engine1 = TTS.GetEngine("ashley","d16");
        TTSEngine engine2 = TTS.GetEngine("james","d16");
        for(int i = 0; i < 100; i++){
            TTSConverter converter = new TTSConverter();
            if((i % 2) == 0){
                converter.Engine = engine1;
            }else{
                converter.Engine = engine2;
            }
            converter.Text = "Hello there.";
            converter.Volume = 250;
            converter.Pitch = 50;
            converter.Speed = 125;
            converter.Pause = 0;
            converter.CommaPause = 0;
            converter.EmphasisFactor = 0;
            converter.TextType = TextType.Normal;
            converter.IsAsync = true;
            ThreadPool.QueueUserWorkItem(converter.ConvertToBufferThreadProc);
        }
    }
}
```

This will queue 100 conversions on the thread pool, 50 using the voice engine ashley/d16 and 50 using the voice engine james/d16. Note that only conversions using different voice engines can run in parallel. For this example, 2 conversions will run in parallel while the remaining will wait in the thread pool until the voice engine's lock is released by the previous conversion.

Android

Voice Engine Installation

Installation of voice engines is not required on Windows and Linux. On Android however, the voice databases will reside inside the compressed .apk file which is built by Unity. This means that the TTS system can not access the database files directly. As a
consequence, running on Android requires that the voice engines are installed in the applications internal storage. This is handled automatically the first time TTS.Init is called on a device. If a large number of voices are used, this could cause the application to load for a long amount of time once this occurs. It is thus recommended to put the call to TTS.Init at an appropriate time, such as during a loading screen.

Voice Engine loading/unloading

Due to how the backend operates on Android, loading and unloading engines are handled automatically upon conversion. The consequence being that TTSEngine.Load() and TTSEngine.Unload() are no-ops on Android.

Namespace Documentation

ReadSpeaker Namespace Reference

Classes

- class TTS

  The core class for using runtime text to speech within Unity.
- class TTSConverter

  Encapsulates the text-to-speech conversion process.
- class TTSEngine

  Represents a voice engine.
- class TTSSpeaker

  Represents a speaking entity.
- class TTSSpeechCharacteristics

  Represents a set of speech characteristics to be used during synthesis.
- class TTSVoicePreset

  A data container for TTSSpeechCharacteristics.

Enumerations

- enum TextType { Normal = 0 , SSML = 128 }

  Determines how the text will be processed during conversion.
- enum OutputFormat { PCM16 = 0 , PCM8 = 1 }

  Determines what audio output format to use during conversion.

Functions

- delegate void OnWordEvent (IntPtr context, int startPos, int endPos, float time)
- delegate void OnVisemeEvent (IntPtr context, short visemeId, float time)
- delegate void OnMarkEvent (IntPtr context, string markName, float time)
- delegate void OnAudioEvent (IntPtr context, byte[ ] audioData, int length)

Enumeration Type Documentation

OutputFormat

`enum ReadSpeaker.OutputFormat`

Determines what audio output format to use during conversion.

Enumerator	
PCM16	16-Bit Linear PCM.
PCM8	8-Bit Linear PCM.

TextType

`enum ReadSpeaker.TextType`

Determines how the text will be processed during conversion.

Enumerator	
Normal	SSML tags will not be processed.
SSML	SSML tags will be processed.

Class Documentation

Events

- static Action onPauseAll

  Invoked by TTS.PauseAll(). Pauses the audio source of every TTSSpeaker.

- static Action onResumeAll

  Invoked by TTS.ResumeAll(). Pauses the audio source of every TTSSpeaker.

- static Action onInterruptAll

  Invoked by TTS.InterruptAll()

Detailed Description

The core class for using runtime text to speech within Unity.

Member Function Documentation

GetAvailableGendersForLanguage()

```
static List< string > ReadSpeaker.TTS.GetAvailableGendersForLanguage (
    string language ) [inline], [static]
```

Gets all of the available genders for a specified language.

Parameters	
Language	The language which is queried.

Returns

A list of genders available for the engines with language language

GetAvailableLanguages()

`static List< string > ReadSpeaker.TTS.GetAvailableLanguages ( ) [inline], [static]`

Gets all available languages in the installed engines.

Returns

All of the languages available from the currently installed engines.

GetDefaultSpeaker()

`static TTSSpeaker ReadSpeaker.TTS.GetDefaultSpeaker ( ) [inline], [static]`

Gets the default speaker or creates one if none exists.

Returns

The current default speaker.

GetEngine()

```
static TTSEngine ReadSpeaker.TTS.GetEngine (
    string name,
    string type ) [inline], [static]
```

Gets the TTSEngine with a specified name and type.

Parameters	
name	The name of the engine.
type	The type of the engine.

Returns

The TTSEngine with a name and type.

GetEngineByID()

```
static TTSEngine ReadSpeaker.TTS.GetEngineByID (
    string engineID ) [inline], [static]
```

Gets the TTSEngine with a specified ID.

Parameters	
EngineID	The ID which is queried.

Returns

The TTSEngine with engineID.

GetEnginesWithGender()

```
static List< TTSEngine > ReadSpeaker.TTS.GetEnginesWithGender (
    string gender ) [inline], [static]
```

Gets all TTSEngines with a specified gender

Parameters	
gender	The gender which is queried.

Returns

The TTSEngines with gender.

GetEnginesWithLanguage()

```
static List< TTSEngine > ReadSpeaker.TTS.GetEnginesWithLanguage (
string language ) [inline], [static]
```

Gets all TTSEngines with a specified language.

Parameters	
language	The language which is queried.

Returns

The TTSEngines with language.

GetEnginesWithLanguageAndGender()

```
static List< TTSEngine > ReadSpeaker.TTS.GetEnginesWithLanguageAndGender (
    string language,
    string gender ) [inline], [static]
```

Gets all TTSEngines with specified language and gender.

parameters	
language	The language which is queried
gender	The gender which is queried.

Returns

The TTSEngines with language and gender.

GetEngineWithGender()

```
static TTSEngine ReadSpeaker.TTS.GetEngineWithGender (
    string gender ) [inline], [static]
```

Get a TTSEngine for a specific gender.

Parameters	
gender	The gender which is queried.

The first found TTSEngine for a gender, null if none is found.

GetEngineWithLanguage()

```
static TTSEngine ReadSpeaker.TTS.GetEngineWithLanguage (
    string language ) [inline], [static]
```

Get a TTSEngine for a specified language.

Parameters	
language	The language which is queried.

Returns

The first found TTSEngine for a language . null if none is found.

GetEngineWithLanguageAndGender()

```
static TTSEngine ReadSpeaker.TTS.GetEngineWithLanguageAndGender (
    string language,
    string gender ) [inline], [static]
```

Get a TTSEngine with a specified language and gender.

Parameters	
language	The language which is queried.

Parameters	
gender	The gender which is queried.

The first found TTSEngine for a language and <paramref="gender">. null if none is found.

GetInstalledEngines()

`static List< TTSEngine > ReadSpeaker.TTS.GetInstalledEngines ( ) [inline], [static]`

Gets all of the installed TTSEngines.

Returns

All installed TTSEngines.

Init()

`static void ReadSpeaker.TTS.Init ( ) [inline], [static]`

Initializes the text to speech system. Has to be called before any calls to associated functions.

Exceptions	
System.Exception	The current platform is not supported.

PauseAll()

`static void ReadSpeaker.TTS.PauseAll( ) [inline], [static]`

Pauses the audio source of every TTSSpeaker.

ResumeAll()

`static void ReadSpeaker.TTS.ResumeAll( ) [inline], [static]`

Resumes the audio source of every TTSSpeaker.

InterruptAll()

`static void ReadSpeaker.TTS.InterruptAll ( ) [inline], [static]`

Interrupts the audio source of every TTSSpeaker.

PlayAudioBuffer()

```
static void ReadSpeaker.TTS.PlayAudioBuffer (
    float[ ] audioData,
    AudioSource audioSource,
    bool playOneShot = false,`
    int sampleRate = 16000 ) [inline], [static]
```

Plays audioData from a buffer on the specified audioSource.

Parameters	
audioData	The buffer which contains the data that is to be played.
audioSource	The AudioSource from which the audio is to be played.

PlayAudioFile()

```
static void ReadSpeaker.TTS.PlayAudioFile (
    string path,
    AudioSource audioSource,
    bool playOneShot = false ) [inline], [static]
```

Plays an audio file located at path on the specified audioSource .

Parameters	
path	The path to the audio file which is to be played.
audioSource	The AudioSource from which the data is to be played.

Say() [1/3]

```
static void ReadSpeaker.TTS.Say (
    string text,
    TextType textType = TextType.Normal,
    bool playOneShot = false ) [inline], [static]
```

Converts a text to speech using the default speaker and immediately plays it.

Parameters	
text	The text which is to be spoken.

Parameters	
textType	The text type which determines how the input text should be processed.
playOneShot	Whether the audio should be played via PlayOneShot.

Say() [2/3]

```
static void ReadSpeaker.TTS.Say (
    string text, TTSSpeaker speaker,
    TextType textType = TextType.Normal,
    bool playOneShot = false ) [inline], [static]
```

Converts a text to speech using the specified speaker and immediately plays it.

Parameters	
text	The text which is to be spoken.
speaker	The speaker who will speak the text.
textType	The text type which determines how the input text should be processed.
playOneShot	Whether the audio should be played via PlayOneShot.

Say() [3/3]

```
static void ReadSpeaker.TTS.Say (
    string text,
    TTSSpeechCharacteristics characteristics,
    AudioSource audioSource,
    TextType textType = TextType.Normal,
    bool playOneShot = false ) [inline], [static]
```

Converts a text to speech using the specified characteristics and immediately plays it from an audioSource.

Parameters	
text	The text which is to be spoken.
characteristics	The speech characteristics which are to be used during synthesis.
audioSource	The AudioSource from which the audio is to be played.
textType	The text type which determines how the input text should be processed.
playOneShot	Whether the audio should be played via PlayOneShot.

SayAsync() [1/3]

```
static void ReadSpeaker.TTS.SayAsync (
    string text,
    TextType textType = TextType.Normal,
    bool playOneShot = false ) [inline], [static]
```

Converts a text to speech asynchronously using the default speaker and plays it when ready.

Parameters	
text	The text which is to be spoken.
textType	The text type which determines how the input text should be processed.
playOneShot	Whether the audio should be played via PlayOneShot.

SayAsync() [2/3]

```
static void ReadSpeaker.TTS.SayAsync (
    string text,
    TTSSpeaker speaker,
    TextType textType = TextType.Normal,
    bool playOneShot = false ) [inline], [static]
```

Converts a text to speech asynchronously using the specified speaker and plays it when ready

Parameters	
text	The text which is to be spoken.
speaker	The speaker who will speak the text.
textType	The text type which determines how the input text should be processed.
playOneShot	Whether the audio should be played via PlayOneShot.

SayAsync() [3/3]

```
static void ReadSpeaker.TTS.SayAsync (
    string text,
    TTSSpeechCharacteristics characteristics,
    AudioSource audioSource,
    TextType textType = TextType.Normal,
    bool playOneShot = false,
    MonoBehaviour monoBehaviour = null ) [inline], [static]
```

Converts a text to speech asynchronously using the specified characteristics and plays it from an audioSource when ready.

Parameters	
text	The text which is to be spoken.
characteristics	The speech characteristics which are to be used during synthesis.
audioSource	The AudioSource from which the audio is to be played.
textType	The text type which determines how the input text should be processed.
playOneShot	Whether the audio should be played via PlayOneShot.

ReadSpeaker.TTSConverter Class Reference

Public Attributes

- OnWordEvent onWord
- OnVisemeEvent onViseme
- OnMarkEvent onMark
- OnAudioEvent onAudio

Properties

- string Text [getset]

  Gets or sets the text to be converted.

- TTSEngine Engine [getset]

  Gets or sets the voice engine to be used for synthesis.

- int Volume [getset]

  Gets or sets the volume to be used during synthesis.

- int Pitch [getset]

  Gets or sets the pitch to be used during synthesis.

- int Speed [getset]

  Gets or sets the speed to be used during synthesis.

- int Pause [getset]

  Gets or sets the time in milliseconds to pause when encountering a delimiter during synthesis.

- int CommaPause [getset]

  Gets or sets the time in milliseconds to pause when encountering a comma during synthesis.

- TextType TextType [getset]

  Gets or sets the text type to be used during synthesis.

- string OutputPath [getset]

  Gets or sets the path to the output file when converting to file.

- OutputFormat OutputFormat [getset]

  Gets or sets the format of the audio output.

- bool IsAsync [getset]

  Gets or sets a value indicating whether the converter is used asynchronously.

Detailed Description

Encapsulates the text-to-speech conversion process.

Member Function Documentation

ConvertToBuffer()

`int ReadSpeaker.TTSConverter.ConvertToBuffer ( ) [inline]`

Converts text to speech using the current handle values and stores the result in an audio buffer.

For example:

```
public class TTSExample : MonoBehaviour {
    public void Start() {
        TTS.Init();
        TTSEngine engine = TTS.GetEngine("ashley","d16");
        TTSConverter converter = new TTSConverter();
        converter.Text = "Hello";
        converter.Engine = engine;
        converter.Volume = 225;
        converter.Pitch = 125;
        converter.Speed = 125;
        converter.Pause = 0;
        converter.CommaPause = 0;
        converter.ConvertToBuffer();
        float[] audioData = converter.GetAudioData();
    }
}
```

Results in audioData containing the audio data from converting the text "Hello" to speech using the speech engine named "ashley" with type "d16".

Returns

0 if successful, a number less than 0 if unsuccessful.

ConvertToBuffer_SyncInfoThreadProc()

```
void ReadSpeaker.TTSConverter.ConvertToBuffer_SyncInfoThreadProc (
    System.Object threadContext ) [inline]
```

A thread procedure to convert text to speech with additional synchronization info using the current handle values and stores the result in an audio buffer. Use this for thread safe conversion.

Parameters	
threadContext	The thread context where this procedure performs the task.

ConvertToBufferThreadProc()

```
void ReadSpeaker.TTSConverter.ConvertToBufferThreadProc (
    System.Object threadContext ) [inline]
```

A thread procedure to convert text to speech using the current handle values and stores the result in an audio buffer. Use this for thread safe conversion.

Parameters	
threadContext	The thread context where this procedure performs the task.

ConvertToFile()

`int ReadSpeaker.TTSConverter.ConvertToFile ( ) [inline]`

Converts text to speech using the current handle values and stores the result in an audio file.

For example:

```
public class TTSExample : MonoBehaviour{
    public void Start(){
        TTS.Init();
        TTSEngine engine = TTS.GetEngine("ashley","d16");
        TTSConverter converter = new TTSConverter();
        converter.Text = "Hello";
        converter.Engine = engine;
        converter.Volume = 225;
        converter.Pitch = 125;
        converter.Speed = 125;
        converter.Pause = 0;
        converter.CommaPause = 0;
        converter.OutputPath = Application.dataPath + "/Hello.wav";
        converter.ConvertToFile();
    }
}
```

Results in a file named 'Hello.wav' being created at Application.dataPath containing the speech output from converting the text "Hello" with the engine named "ashley" with type "d16"

Returns

0 if successful, a number less than 0 if unsuccessful.

ConvertToFileThreadProc()

```
void ReadSpeaker.TTSConverter.ConvertToFileThreadProc (
    System.Object threadContext ) [inline]
```

A thread procedure to convert text to speech using the current handle values and stores the result in an audio file. Use this for thread safe conversion.

Parameters	
threadContext	The thread context where this procedure performs the task.

FinishedConverting()

`bool ReadSpeaker.TTSConverter.FinishedConverting ( ) [inline]`

Check if the conversion has finished.

Returns

True if conversion has both been started and finished, false otherwise.

GetAudioData()

`float[ ] ReadSpeaker.TTSConverter.GetAudioData ( ) [inline]`

Gets the audio data that has been converted by ConvertToBuffer().

Returns

The audio data which has been converted by ConvertToBuffer(). The complete data set if FinishedConverting() returns true, otherwise an incomplete data set.

ReadSpeaker.TTSEngine Class Reference

Represents a voice engine.

Public Attributes

- readonly string id

  The ID of the engine.
- readonly string name

  The name of the engine.
- readonly string type

  The type of the engine.
- readonly string language

  The language used by the voice.
- readonly string gender

  The gender of the voice.
- readonly string version

  The Version number of this engine.
- readonly int sampleRate

  The sample rate used by the voice.

Detailed Description

Represents a voice engine.

Member Function Documentation

Equals()

```
bool ReadSpeaker.TTSEngine.Equals (
    TTSEngine other ) [inline]
```

Compare this instance to another TTSEngine instance.

Parameters	
other	The TTSEngine to compare to.

Returns

True if this voice engine has the same ID as others. False otherwise.

ReadSpeaker.TTSSpeaker Class Reference

Represents a speaking entity.

Public Attributes

- TTSSpeechCharacteristics characteristics

  The speech characteristics of this speaker.
- TTSVoicePreset preset

  The voice preset of this speaker.
- bool usePreset

  Whether to use the voice preset or the inherent speech characteristics.
- AudioSource audioSource

  The audio source used by this speaker.

Detailed Description

Represents a speaking entity

Member Function Documentation

GetSpeechCharacteristics()

```
TTSSpeechCharacteristics ReadSpeaker.TTSSpeaker.GetSpeechCharacteristics ( ) [inline]
```

Gets the speech characteristics which are currently in use by this speaker.

Returns

If usePreset is set to true, it returns the characteristics defined by preset . Otherwise returns the inherent characteristics.

ReadSpeaker.TTSSpeechCharacteristics Class Reference

Represents a set of speech characteristics to be used during synthesis.

Public Member Functions

- TTSSpeechCharacteristics (TTSEngine engine)

Parameters	
engine	The voice engine to be used for synthesis.

- TTSSpeechCharacteristics (TTSEngine engine, int volume, int pitch, int speed, int pause, int commaPause)

Properties

- TTSEngine Engine [getset]

  Gets or sets the voice engine to be used for synthesis.

- int Volume [getset]

  Gets or sets the volume to be used during synthesis.

- int Pitch [getset]
- int Speed [getset]
- int Pause [getset]
- int CommaPause [getset]

Detailed Description

Represents a set of speech characteristics to be used during synthesis.

Constructor & Destructor Documentation

TTSSpeechCharacteristics()

```
ReadSpeaker.TTSSpeechCharacteristics.TTSSpeechCharacteristics (
    TTSEngine engine,
    int volume,
    int pitch,
    int speed,
    int pause,
    int commaPause ) [inline]
```

Parameters	
engine	The voice engine to be used for synthesis.
volume	The volume to be used during synthesis.
pitch	The pitch to be used during synthesis.
speed	The speed to be used during synthesis.
pause	The time in milliseconds to pause when encountering a delimiter during synthesis.
commaPause	The time in milliseconds to pause when encountering a comma during synthesis.

ReadSpeaker.TTSVoicePreset Class Reference

Public Attributes

- TTSSpeechCharacteristics characteristics

  The speech characteristics of this preset.

Detailed Description

A data container for TTSSpeechCharacteristics.

Contact

tts_support@readspeaker.com