RT-Voice PRO

Hearing is understanding

Documentation

crosstales LLC

Date: 26. October 2016
Version: 2.5.2

Table of Contents

Thank you for buying our asset "RT-Voice PRO"!

If you have any questions about this asset, send us an email at assets@crosstales.com. Please don't forget to rate it or write a little review – it would be very much appreciated.

1. Overview

Have you ever wanted to make software for people with **visual impairment** or who have **difficulties reading**? Do you have lazy **players** who **don't like to read** too much? Or do you even want to **test** your game's **voice dialogues** without having to pay a voice actor yet? With RT-Voice this is very easily done – it's a major time saver!

RT-Voice uses the computer's (already implemented) TTS (text-to-speech) voices to turn the **written lines** into **speech** and dialogue at **run-time**! Therefore, **all text** in your game/app can be **spoken** out loud to the player.

And all of this without any intermediate steps: The transformation is **instantaneous** and **simultaneous** (if needed)!

2. Features

- **Instantaneous transformation** form text-to-speech! No intermediate steps.
- Since the audio is generated during run-time, it **saves** a lot of **space**!
- **No need** to **voice act** for yourself during the test-phase of your game.
- **Multiple** voices at **once** (e.g. scenes at a public square with many people talking **simultaneously**).
- Fine-tune your voices with **rate**, **pitch** and **volume**
- **Current word**, **visemes** and **phomenes** on Windows and iOS (incl. mark functions)
- Generated audio can be **stored** to **files**. Those files can be **reused** inside **Unity**
- 1-n synchronized **loudspeakers** for a **single** AudioSource **origin**.
- Simple **sequence** and **dialog** system
- **No** performance **overhead**!
- Powerful **API t**o get **maximum control** as a developer
- Developed for **Unity 5**
- Runs on the **Mac-** and **Windows-Standalone** (works with all **SAPI5** compatible voices**)**, **iOS** and **Android** platforms
- **PlayMaker** actions!
- **Test-Drive** the voices inside the **Editor**!
- Extensive **demo** scenes, **documentation**, **API** and **support**!
- Full **C# source code**
- We are **committed** to all our assets! This means, we will add new features over time!

2.1. Supported third-party assets

- SALSA
- Localized Dialogs & Cutscenes (LDC)
- Dialogue System for Unity
- THE Dialogue Engine
- PlayMaker
- Adventure Creator
- LipSync
- SLATE
- Cinema Director
- uSequencer
- Quest System Pro
- NPC Chat

2.2. Platform-specific limitations

2.2.1. Windows

- Native pitch has no effect
- Native rate is internally limited to 20 logarithmic distributed steps
- .NET 4.0 or higher must be installed

2.2.2. MacOS

- Native pitch has no effect
- Native volume has no effect
- No current words, phonemes and visemes

2.2.3. Android

- Only one native voice at the time (can be solved by generating audio)
- No current words, phonemes and visemes
- Minimum Android version: 4.0.3 (API 15)

2.2.4. iOS

- Only one active native voice at the time
- No audio generation
- Current word but no phonemes and visemes
- Minimum iOS-version: 7.1

3. Demonstration

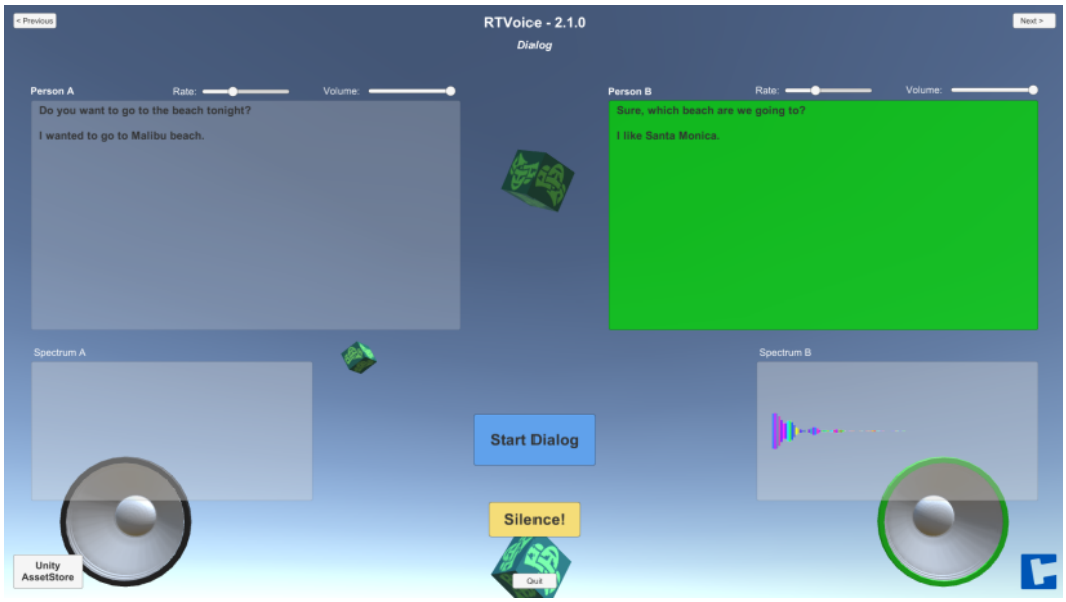
The asset comes with many demo scenes to show the main usage.

3.1. Speech

This demo scene shows how to transform written lines into speech. Choose your preferred voice.

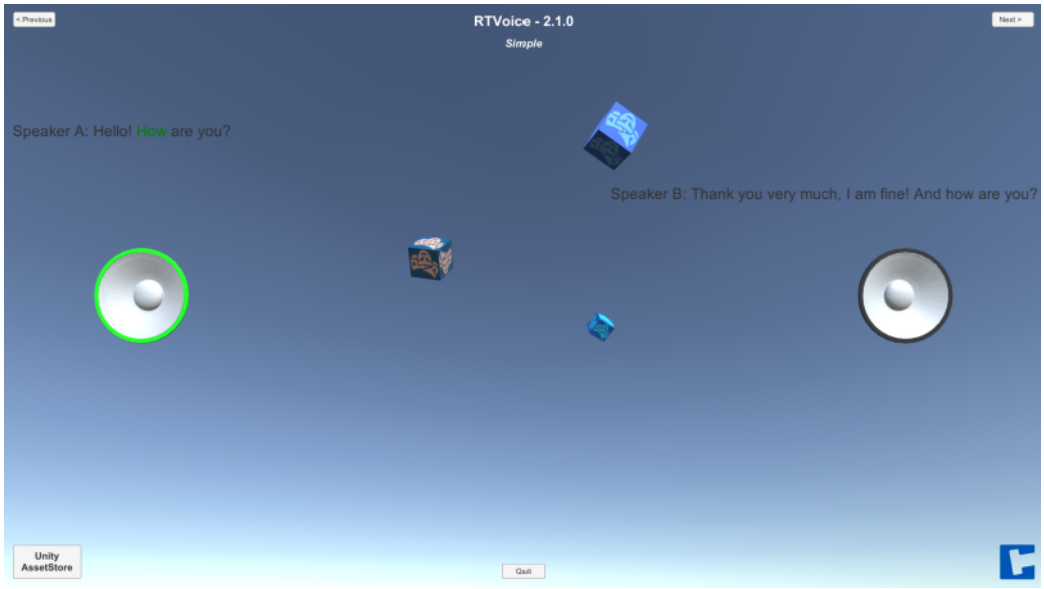
3.2. Dialog

In this demo scene you can act out a dialogue between two "people". You can choose a different voice for both participants.



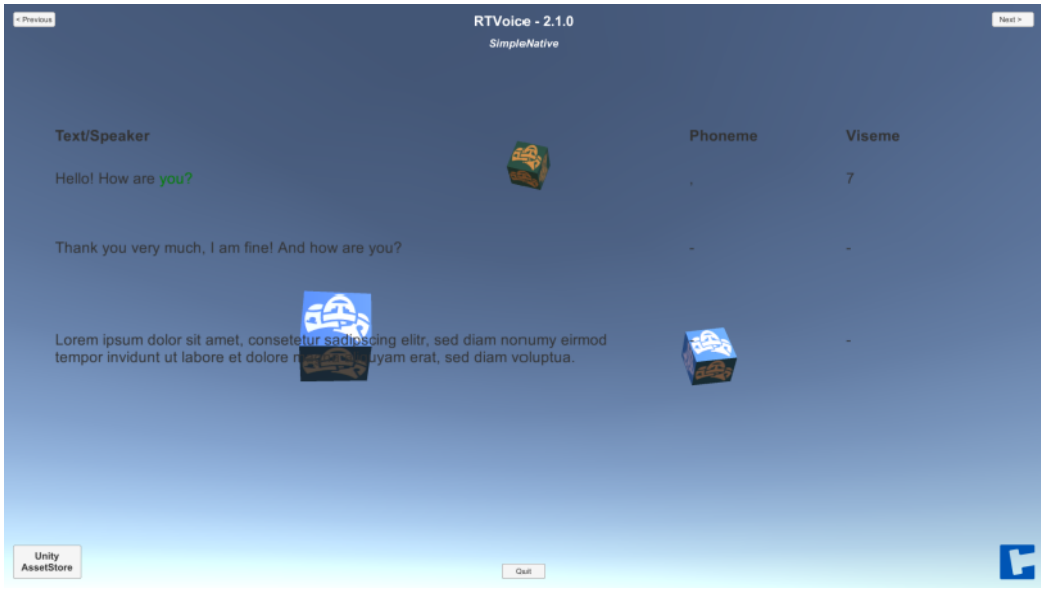
3.3. Simple

The "Simple" scene shows the easiest and recommended way for most purposes with generated audio.



3.4. SimpleNative

The "SimpleNative" scene shows the easiest way for native audio.



3.5. 3DAudio

This scene demonstrates 3D positioned and looped audio.

Needs the Unity Standard Characters (Assets → Import Packages → Characters).

3.6. Loudspeakers

This scene demonstrates 3D positioned loudspeakers with only one audio origin (looped). Needs the Unity Standard Characters (Assets → Import Packages → Characters).

3.7. SendMessage

This scene shows the usage of Unity's "SendMessage".

3.8. Sequencer

This scene shows the usage of our simple sequencer.

3.9. Native and PreGenerated

These two scenes are showing how you can build applications with exact timing between audio and animations (e.g. lip sync).

3.10. SpeechText

This scene shows how to speak or store generated audio (see the result inside the folder "_generatedAudio").

4. Setup

RT-Voice has global settings under "Edit\Preferences..." and under "Tools\RTVoice\Configuration...".

4.1. Add RT-Voice

Normally, RT-Voice is added automatically to your scene. To add RT-Voice manually to your project, there are two ways:

1. Add the prefab **RTVoice** to the scene
2. Or go to *Tools => RTVoice => Add => **RTVoice***

4.2. Add SpeechText

To add a SpeechText to your project, there are two ways:

1. Add the prefab **SpeechText** to the scene
2. Or go to *Tools => RTVoice => Add => **SpeechText***

4.3. Add Sequencer

To add a Sequencer to your project, there are two ways:

1. Add the prefab **Sequencer** to the scene
2. Or go to *Tools => RTVoice => Add => **Sequencer***

4.4. Add Loudspeaker

To add a Loudspeaker to your project, there are two ways:

1. Add the prefab **Loudspeaker** to the scene
2. Or go to *Tools => RTVoice => Add => **Loudspeaker***

4.5. Differences between standard and native mode

In the **standard** mode the TTS-system of your OS will **convert** your text to an **audio** file and return it to **Unity** as an "**AudioSource**" for further use (like changing the volume, pitch etc.).

On the other hand, the **native** mode **delegates** the speech-task **entirely** to the underlying TTS-system (outside of Unity). You are **losing** some **control** but it uses slightly **less performance**.

We clearly **recommend** using the **standard** mode.

4.6. Speaker.cs vs. LiveSpeaker.cs

"Speaker.cs" is the main class of "RT-Voice" and presents the API via static methods. "LiveSpeaker.cs" on the other hand is a wrapper for "Speaker.cs" and presents the API as normal C#-instance via public methods. The main usage of "LiveSpeaker.cs" is as a receiver for "SendMessage"-calls.

5. API

The asset contains various methods and the most important are explained here.

Make sure to **include** the **name space** in your relevant source files:

```
using Crosstales.RTVoice;
```

5.1. Speaker

The "Speaker.cs" is a singleton and contains the following static methods.

5.1.1. Speak

Speaks a text with a given voice and optional AudioSource.

For example:

```
//Immediately speak "hello world" with the first available voice
Speaker.Speak("hello world", audioSource);

//Immediately speak "hello world" with the first English voice (if available else it uses the first voice on your OS)
Speaker.Speak("hello world", audioSource, Speaker.VoiceForCulture("en"));

// Prepare speak "hello world" with the first available voice (without AudioSource.Play() - this is up to you). With this technique, you can prepare all audio texts of a scene and you can modify the AudioSource as you like!
Speaker.Speak("hello world", audioSource, null, false);
```

5.1.2. SpeakNative

Speaks a text with a given voice.

For example:

```
//Speak "hello world" with the first available voice
Speaker.SpeakNative("hello world");

//Speak "hello world" with the first English voice (if available else it uses the first voice on your OS)
Speaker.SpeakNative("hello world", Speaker.VoiceForCulture("en"));
```

5.1.3. Silence

Silence all active TTS-voices.
Example:

```
//Silence all voices
Speaker.Silence();
```

5.1.4. Voices

`List<Voice> Voices`
Returns all available voices (alphabetically ordered by 'Name').

5.1.5. VoicesForCulture

`List<Voice> VoicesForCulture(string culture)`
Returns all available voices for a given culture (alphabetically ordered by 'Name').

5.1.6. VoiceForCulture

`Voice VoiceForCulture(string culture, int index)`
Returns the voice for the given culture and index.

5.1.7. VoiceForName

`Voice VoiceForName(string name)`
Returns the voice for the given name or null if not found.

5.1.8. Cultures

`List<string> Cultures`
Returns all available cultures (alphabetically ordered by 'Culture').

5.2. Callbacks

There are various callbacks available. Subscribe them in the "Start"-method and unsubscribe in "OnDestroy".

5.2.1. Speak start and complete (native)

```
SpeakNativeStart(SpeakEventArgs e);
SpeakNativeStart OnSpeakNativeStart;

SpeakNativeComplete(SpeakEventArgs e);
SpeakNativeComplete OnSpeakNativeComplete;
```

5.2.2. Current word (native and Windows only)

```
SpeakNativeCurrentWord(CurrentWordEventArgs e);
SpeakNativeCurrentWord OnSpeakNativeCurrentWord;
```

5.2.3. Current phoneme (native and Windows only)

```
SpeakNativeCurrentPhoneme(CurrentPhonemeEventArgs e);
SpeakNativeCurrentPhoneme OnSpeakNativeCurrentPhoneme;
```

5.2.4. Current viseme (native and Windows only)

```
SpeakNativeCurrentViseme(CurrentVisemeEventArgs e);
SpeakNativeCurrentViseme OnSpeakNativeCurrentViseme;
```

5.2.5. Speak start and complete

```
SpeakStart(SpeakEventArgs e);
SpeakStart OnSpeakStart;

SpeakComplete(SpeakEventArgs e);
SpeakComplete OnSpeakComplete;
```

5.2.6. Speak audio generation start and complete

```
SpeakAudioGenerationStart(SpeakEventArgs e);
SpeakAudioGenerationStart OnSpeakAudioGenerationStart;

SpeakAudioGenerationComplete(SpeakEventArgs e);
SpeakAudioGenerationComplete OnSpeakAudioGenerationComplete;
```

5.2.7. Errors

```
ErrorInfo(string info);
ErrorInfo OnErorInfo;
```

5.2.8. Example

Get informed when a native speak starts and completes:

```
void Start() {
 // Subscribe event listeners
 Speaker.OnSpeakNativeStart += speakNativeStartMethod;
 Speaker.OnSpeakNativeComplete += speakNativeCompleteMethod;

 Speaker.SpeakNative("Hello world!");
}

void OnDestroy() {
 // Unsubscribe event listeners
 Speaker.OnSpeakNativeStart -= speakNativeStartMethod;
 Speaker.OnSpeakNativeComplete -= speakNativeCompleteMethod;
}

private void speakNativeStartMethod(SpeakNativeEventArgs e) {
 Debug.Log("speakNativeStartMethod: " + e);
}

private void speakNativeCompleteMethod(SpeakNativeEventArgs e) {
 Debug.LogWarning("speakNativeCompleteMethod: " + e);
}
```

5.3. Complete API

For more details, please see the RTVoice-api.pdf

6. Additional voices

RT-Voice works great with third-party voices (e.g. IVONA, Cereproc etc.).

6.1. Windows

All SAPI5-compatible voices are supported. Microsoft also provides a wide range of voices for different languages:

https://www.microsoft.com/en-us/download/details.aspx?id=27224

To install and use those voices follow this manual:

http://superuser.com/a/872573

6.2. MacOS

Apple delivers many voices for different languages. To add or customize them, follow the tutorial below:

http://osxdaily.com/2011/07/25/how-to-add-new-voices-to-mac-os-x/

6.3. Android

You can add various voices on your Android phone:

http://www.geoffsimons.com/2012/06/7-best-android-text-to-speech-engines.html

There is also a pssibility to download high quality voices:

http://www.androidauthority.com/google-text-to-speech-engine-659528/

6.4. iOS

You can only change the quality of the installed voices:

https://support.apple.com/en-us/HT202362

7. Third-party support (PlayMaker etc.)

„RT-Voice" supports various assets from other publishers. Please import the desired packages from the „3rd party"-folder.

8. Upgrade to new version

Follow this steps to upgrade your version of "RT-Voice PRO":

1. Update "RT-Voice PRO" to the latest version from the "Asset Store"
2. Inside your project in Unity, go to menu "File" => "New Scene"
3. Delete the "crosstales\RTVoice" folder from the Project-view
4. Import the latest version from the "Asset Store"

9. Important notes

After this setup, the "RT-Voice" is ready to use. It is important to know that it uses the **singleton**-pattern, which means that **once instantiated**, the "RT-Voice" will **live until** the application is **terminated**.

Remember: it must be instantiated before you try to access it! Otherwise it's not possible to use it.

10. Problems, improvements etc.

If you encounter any problems with this asset, just send us an email with a problem description and the invoice number and we will try to solve it.

11. Release notes

See "README.txt".

12. Contact and further information

crosstales LLC
Weberstrasse 21
CH-8004 Zürich

Homepage:	http://www.crosstales.com/en/assets/rtvoice
Email:	assets@crosstales.com
AssetStore:	https://www.assetstore.unity3d.com/#!/content/41068
Forum:	http://forum.unity3d.com/threads/coming-soon-rt-voice.340046/
Documentation:	http://www.crosstales.com/en/assets/rtvoice/RTVoice-doc.pdf
API:	http://www.crosstales.com/en/assets/rtvoice/api
Windows-Demo:	http://www.crosstales.com/en/assets/rtvoice/RTVoice_demo_win.zip
Mac-Demo:	http://www.crosstales.com/en/assets/rtvoice/RTVoice_demo_mac.zip

13. Our other products

 **Bad Word Filter**	The "Bad Word Filter" (aka profanity or obscenity filter) is exactly what the title suggests: a tool to filter swearwords and other "bad sentences".
 **DJ**	DJ is a player for external music-files. It allows a user to play his own sound inside any Unity-app. It can also read ID3-tags.
 **Radio**	Have you ever wanted to implement radio stations but don't want (or can't) pay an horrendous amount of money? Whenever you like to provide good sound from famous artists for your games or apps, tune in on one of the uncountable Internet MP3/OGG radio stations available for free.
 **TPS**	Turbo Platform Switch is a Unity editor extension to reduce the time for assets to import during platform switches. We measured speed improvements up to 50x faster than the built-in switch in Unity.