Voice Handler

Overview

The voice handler is basically a simplified way in which to play complex sound events that deal with voices. It is also a way to easily handle subtitles along with the voices and play special effects track along with them. It has a queue system and keeps track of how many times a voice has been played.

Note that while often used along with the event database, these are two completely separate systems, so they should not be confused with one another.

Setup

Before calling a subject the source of the sound should be set up. This is done with SetVoiceSource() in cVoiceHandler, or better the helper function Voice_SetSource(). This sets up the entity (which can be something moving) which will emit the voice coming from the specified character. It will also set up the distance that the voice can be heard. Do this in OnEnter in the map script file.

Properties

Character

The character is the agent that is speaking the lines. Normally this is a person of some sort but it can really be whatever. It mostly used used as a way to give debug data on what character is saying what line, and also used to place the voice in the world (the position of character can be set in script).

ID
Internal Id

Name
Name of the character

EntryType
The type of the sounds (and extrasounds) that are played for this character. Is either World, WorldClean or GUI.

Volume
Volume of the sound files. Gets multiplied with other volume properties.

ReverbVolume
The amount of reverb used for the sounds. Gets multiplied with other ReverbVolume properties.

Scene

This does not have to mean the level that the voice are played in, but what subtext that they are used in. A scene is NOT the same as the level the voices belong to, but a level can (and should have) have many scenes. It is a way to group subjects and also to setup what voices should be in focus (by using FocusPrio).

ID
Internal Id

Name
Name of the scene

FocusPrio
This is the priority for a voice to into focus, meaning that is subtittles are shown and any other voice playing gets its volume lowered. To beat a currently playing voice, FocusPrio must be higher.
When a subject starts playing it is checked if another sound is playing and FocusPrio determines if the new subject gets into focus. Also when a subject stops playing a certain amount of time is waited (default 2 seconds) and then there is a check to see if there is any playing voice that should come into focus. If there are many playing, FocusPrio determines which one. The reason for waiting a litte while is in case there is a conversiation in one scene containing many subjects, and if so it would not sound good if the focus was constantly switched (ie unrelated subtitles popping up for a second).

Volume
Volume of the sound and effect files. Gets multiplied with other volume properties.

ReverbVolume
The amount of reverb used for the sounds. Gets multiplied with other ReverbVolume properties.

Effect

This defines certain effect that should be played on the voice, be that echo, noise or whatnot.
Note: Right now there is not much done with effect, so it is pretty useless

ID
Internal Id

Name
Name of the effect

GlobalStartEffectFile
A file that is played each time a new line starts.

GlobalEndEffectFile
A file that is played each time a new line is over.

GlobalVoiceOffset
Not used.

Subject

A subject is what you call when you want to voice to be played. A subject is the basic data structure of the voice data and it contains sound files, subtitle texts and various options.
A subject contain one or more **Lines** (more on these below). Normally all of these lines will be played in order, but if UseSingleRandomLine is true, only one will.

ID
Internal Id

Name
Name of the subject

UseSingleRandomLine
If only a single random line should be used. If more than 2 lines, it is made sure that the a line is never picked two times in a row.

SceneId
Id of scene connected to the subject.

EffectId
Id of the effect used for the subject.

Line

When a voice is called to be played the entirety of a line is always played. The line contains one or more **Sounds** (more on these below) that make up what is played.

ID
Internal Id

CharacterId
Id of character connected to the line.

Sound

This is the lowest level data structure for a subject. It contains properties for the voice, (opitonal) effect sound, and subtitle.

The file name for the voice sound is generated based on the names of the higher level structures. The syntax is:
`[map]/[scene]_[subject]_[line index in 3 digits]_[character]_[sound index in 3 digits]`

Here is an example:
01_01_castle/SwordFight_ShowMercy_002_Arthur_001
This would be a voice sound being said in the map *"01_01_castle"* (no file extension!), in the Scene

"SwordFight". The subject is *"ShowMercy"*, it is the second line, it is being said by *"Arthur"* and it is his first sound for that line.

The lang file entry for the text is also generated, with this syntax:
Category: Voices_[map]
Entry: [scene]_[subject]_[line index in 3 digits]_[character]_[sound index in 3 digits] (apart from no folder name, it is exactly the same as for the voice file!)

Finally, if "AutogenerateLangFiles" in "Main" is set to true in the user settings, then the lang file entries will be autogenerated with the text specified in the sound's properties.

Text
The text for the voices, not really saved by added to a lang file (if AutogenerateLangFiles is true i settings, see above)

HasVoice
If the sound has a voice at all. This is only meaningful false if there is an Effect filed specificed.

Volume
Volume of the sound and effect file. Gets multiplied with other volume properties.

ReverbVolume
The amount of reverb used for the sounds. Gets multiplied with other ReverbVolume properties.

VoiceOffset
Number of seconds before voice starts to play.

TextOffset
Number of seconds before the subtitles show up.

ExtraEffectOffset
Number of seconds before the effect file starts to play.

ExtraEffectFile
File path to an extra sound file that is played along with the voice.

EndsAfterExtraEffect
If the sound should end when the extra sound file is done playing (and not the voice).

EndPadding
Number of seconds of padding in the length of voice file. If this is over 0, then that means the next sound will start that many seconds earlier (while the current sound is still playing).

Use with AI

When using voice with AI it is best always through the BarkMachine component using `BarkMachine_PlayVoice(…)`. That way it is sure to be played in the right place even if there are multiple agents that use voices from the same character. When using the BarkMachine, the source set up directly before the line is played.

The voices for all agents should also have “AgentBark” as scene, this way it makes it possible to see to that there is only one dialog line from an Agent at a time. Should there be some special line that does not apply to this, then by all means do not do this.

If you want to have longer specific dialog on agents, then the best way to set that up is to have the piece of a dialog as a single subject with two unique Characters attached to the lines. Then set up these characters as voice sources specifically in the script and start the dialog using `Voice_Play`. The scene must be AgentBark as described above and the prio must be less than what the default barks use. What then happens is that the dialog will continue until some AI event triggers default sounds to be played and the specific dialog will automatically be stopped.

Best Practices

Here are a few practices that should be used when adding voices in the game:

- Sound effects are between *:s
When adding sound effects, like *gasp*, *pants*, *cries*, etc then always put them between *:s. This way we can easily remove them for people that do not want the hearing impaired option.

- Pure sound effects is a separate line
When implementing a subject that contains parts where there is only (non character related) sound effects playing you shall add this as Line on its own. The character containing it shall be called “_SoundEffect”. If the sound effects need to come from special position in the world, then do add a “_” followed by a prefix to the name, for instance “_SoundEffect_Trees” (this should be pretty rare though and these sort of effects are better added through callback functions, but there might be some rare instance that require it).

Important note is that this only meant for sound effects that are not specific to a character voice. Examples:
Do **NOT** use it for: *cough*, *clears throat*, *spits*, etc.
DO use it for: *steps are heard coming down the stairs*, *large metal door opens*, *Anne carefully steps on the creaking floor*, etc.

- All protagonist dialog come from the character “Player”
Make sure that all the dialog that the player is supposed to say outloud comes from the character “Player”, as this allow the game to not play breathing, etc when a voice over from the player character is playing.

- Use proper prefix for subjects
Have prefixes on the subjects that give some hint where they belong and make it easier to sort. For instance a blackbox subject should start with “Blackbox_”, eg “Blackbox_FriedRobot”. Another examples would be a situation that many subjects belong. Say Simon meets a farmer and there is a lot of subjects related to this, then use the prefix “SimonMeetsFarmer_”, eg “SimonMeetsFarmer_Intro”.

This is also true for special characters, for instance when a certain character speak in a recording. So for instnace a character that appears in a SOMA blackbox message (that does not come from a world position), should have “Blackbox_” as prefix, eg “Blackbox_Amy”.

- Write temp dialog in square brackets

If you add some writing that is only temporary then add this between square brackets. Also briefly summarize what is supposed to be going on, or what the is supposed to be expressed. Examples:
[Player gets his finger caught in bear trap]
[Hint that the screw needs to be bigger]

If you need to have further info that feels wrong to show in-game then you can add extra info under the line direction, but make sure to start the line with INFO:, so it is not confused with proper line direction. Examples:
INFO: The player tries to hold the pain inside.
INFO: The solution is to use the file on the screw, can have slight hint for that.

File Format

```
<VoiceData>
  <Characters>
    <Character [Properties] />
    ...
  </Characters>
  <Effects>
    <Effect [Properties] />
    ...
  </Effects>
  <Scenes>
    <Scene [Properties] />
    ...
  </Scenes>
  <Subjects>
    <Subject [Properties] >
      <Line [Properties] >
        <Sound [Properties] />
        ...
      </Line>
      ...
    </Subject>
  </Subjects>
</VoiceData>
```

For information on the Properties that can be used, see Properties section above.