helper_audio.hps

Helper functions for handling sound, music and dialogue

_Sound_Global_EntryAffected

```
bool _Sound_Global_EntryAffected(int alIdx,
                                 eSoundEntryType aAffectedTypes)
```

Sound_SetGlobalSpeed

```
void Sound_SetGlobalSpeed(float afFreq,
                          eSoundEntryType
aAffectedTypes=eSoundEntryType_WorldAll)
```

Sets the relative frequency of all sounds.

- **afFreq**: the relative frequency to use. 1= normal, 2=doubled, 0.5=half as fast, etc
- **aAffectedTypes**: The kind of sounds affected, possible values: eSoundEntryType_World, eSoundEntryType_WorldClean, eSoundEntryType_WorldAll, eSoundEntryType_Gui or eSoundEntryType_All

---

Sound_FadeGlobalSpeed

```
void Sound_FadeGlobalSpeed(float afFreq,
                           float afTime,
                           eSoundEntryType
aAffectedTypes=eSoundEntryType_WorldAll)
```

Fade the relative frequency of all sounds.

- **afFreq**: the relative frequency to use. 1= normal, 2=doubled, 0.5=half as fast, etc
- **afTime**: the time which the fade takes
- **aAffectedTypes**: The kind of sounds affected, possible values: eSoundEntryType_World, eSoundEntryType_WorldClean, eSoundEntryType_WorldAll, eSoundEntryType_Gui or eSoundEntryType_All

---

Sound_SetGlobalVolume

```
void Sound_SetGlobalVolume(float afVolume,
                           eSoundEntryType
aAffectedTypes=eSoundEntryType_WorldAll)
```

Set the relative volume of all sounds.

- **afVolume**: the relative volume to use. possible values: 0 - 1
- **aAffectedTypes**: The kind of sounds affected, possible values: eSoundEntryType_World, eSoundEntryType_WorldClean, eSoundEntryType_WorldAll, eSoundEntryType_Gui or eSoundEntryType_All

---

Sound_FadeGlobalVolume

```
void Sound_FadeGlobalVolume(float afVolume,
                            float afTime,
                            eSoundEntryType
aAffectedTypes=eSoundEntryType_WorldAll)
```

Fade the relative volume of all sounds.

- **afVolume**: the relative volume to use. possible values: 0 - 1
- **afTime**: the time which the fade takes
- **aAffectedTypes**: The kind of sounds affected, possible values: eSoundEntryType_World, eSoundEntryType_WorldClean, eSoundEntryType_WorldAll, eSoundEntryType_Gui, , eSoundEntryType_GuiWorld or eSoundEntryType_All

---

Sound_CreateAIEventAtEntity

```
void Sound_CreateAIEventAtEntity(const tString &in asEntity,
                                 float afRadius,
                                 int alPrio)
```

Creates an event that is heard by any entity with the Listener component

- **asEntity**: Name of the entity to create it at. Wildcardds (*) possible.
- **afRadius**: How big the event's sphere of influence will be.
- **alPrio**: The prio of the event. Some listener sort out certain prios.

Sound_CreateAtEntity

```
void Sound_CreateAtEntity(const tString &in asSoundName,
                          const tString &in asSoundFile,
                          const tString &in asEntity,
                          float afFadeTime=0.0f,
                          bool abSaveSound=false,
                          float afTargetVolume=1.0f)
```

Create and play a sound (.snt) file at the position of an entity or player.

- **asSoundName**: name of the sound that is used when you like to, for example, stop it.
- **asSoundFile**: name of the FMOD event or .snt sound file to play. FMOD always needs to be full path ie “project/event_group(s)/event”.
- **asEntity**: name of the entity to play the sound at, can also be “player”.
- **afFadeTime**: fade in the sound during this many seconds, 0 = no fade.
- **abSaveSound**: if the sound should be saved, ie if sound loops and the sound should play on exit and enter of the level.
- **afTargetVolume**: the volume (0.0 - 1.0) the sound should be played at. Defaults to 1.0f.

Sound_CreateAtEntity_UsePrefix

```
void Sound_CreateAtEntity_UsePrefix(const tString &in asSoundName,
                                    const tString &in asSoundFile,
                                    const tString &in asEntity,
                                    float afFadeTime,
                                    bool abSaveSound,
                                    float afTargetVolume=1.0f)
```

Create and play a sound (.snt) file at the position of an entity or player. If a Soundscape area SoundPrefix is set, then that will be used as a parent folder for the sound.

- **asSoundName**: name of the sound that is used when you like to, for example, stop it.
- **asSoundFile**: name of the FMOD event or .snt sound file to play. FMOD always needs to be full path ie “project/event_group(s)/event”. If the prefix is not “”, then the path is “project/event_group(s)/prefix/event”.
- **asEntity**: name of the entity to play the sound at, can also be “player”.
- **afFadeTime**: fade in the sound during this many seconds, 0 = no fade.
- **abSaveSound**: if the sound should be saved, ie if sound loops and the sound should play on exit and enter of the level.
- **afTargetVolume**: the volume (0.0 - 1.0) the sound should be played at. Defaults to 1.0f.

Sound_Play

```
void Sound_Play(const tString &in asSoundName,
                float afFadeTime,
                bool abResetVolMul=false)
```

Play a sound that exists in the level editor, attached to an entity, or previously created that has been stopped and now should be played again.

- **asSoundName**: name of the sounds that you would like to play. Wildcard(s) * are supported.

- **afFadeTime**: fade in the sound during this many seconds, 0 = no fade.
- **abResetVolMul**: reset the FadeVolumeMul set when using Sound_Fade.

Sound_Stop

```
void Sound_Stop(const tString &in asSoundName,
                float afFadeTime)
```

Stop a sound that is currently playing. Can be sound entities or Gui sounds.

- **asSoundName**: name of the sound that you would like to stop. Wildcard(s) * are supported.
- **afFadeTime**: fade out the sound during this many seconds, 0 = no fade. Must be 0 if using a FMOD sound event with Sustain Point.

Sound_Fade

```
void Sound_Fade(const tString &in asSoundName,
                float afVolumeDest,
                float afFadeTime)
```

Fade/Change the volume of a sound entity.

- **asSoundName**: the sound file to fade. Wildcard(s) * are supported.
- **afVolumeDest**: at what relative volume to fade the sound to. 1 = default volume of entity, can't go over 1.
- **afFadeTime**: how long in seconds the fade should take.

Sound_PlayGui

```
void Sound_PlayGui(const tString &in asSoundFile,
                   float afVolume,
                   eSoundEntryType aEntryType=eSoundEntryType_Gui)
```

Play a sound without any world position, either .snt files or any of the supported formats (.wav, .ogg etc) without the need of a .snt file.

- **asSoundFile**: the sound file to play.
- **afVolume**: at what volume to play the sound.
- **aEntryType**: the entry type of the sound.

Sound_FadeInGui

```
void Sound_FadeInGui(const tString &in asSoundFile,
                     float afVolume,
                     float afFadeTime)
```

Fades in a GUI sound. This is can called directly after Sound_PlayGui or in case the sound doesn't exist, start it.

- **asSoundFile**: the sound file for the sound that is playing
- **afVolume**: volume to fade in to.
- **afFadeTime**: Time it takes to fade it out

Sound_StopGui

```
void Sound_StopGui(const tString &in asSoundFile,
                   float afFadeTime,
                   bool abPlayEnd=true)
```

Stops a GUI sounds

- **asSoundFile**: the sound file for the sound that is playing
- **afFadeTime**: Time it takes to fade it out
- **abPlayEnd**: If the end (setup in the event) should be played. Only applicable is afFadeTime is 0

Sound_FadeGuiVolume

```
void Sound_FadeGuiVolume(const tString &in asSoundFile,
                         float afVolumeDest,
                         float afFadeTime)
```

Fades a GUI sound volume to another

- **asSoundFile**: the sound file for the sound that is playing
- **afVolumeDest**: The new volume (note, if 0 it will not stop!)
- **afFadeTime**: Time the fade takes

Sound_FadeGuiSpeed

```
void Sound_FadeGuiSpeed(const tString &in asSoundFile,
                        float afSpeedDest,
                        float afFadeTime)
```

Fades a GUI sound speed to another

- **asSoundFile**: the sound file for the sound that is playing
- **afSpeedDest**: The new speed (note, if 0 it will not stop!)
- **afFadeTime**: Time the fade takes

Sound_SetGuiParam

```
void Sound_SetGuiParam(const tString &in asSoundFile,
                       const tString &in asParam,
                       float afValue)
```

Sets the param of a gui sound

- **asSoundFile**: the sound file for the sound that is playing
- **asParam**: The param variable name
- **afValue**: The new value to give it.

Sound_GuiIsPlaying

```
bool Sound_GuiIsPlaying(const tString &in asSoundFile)
```

Checks if a gui sound is playing.

- **asSoundFile**: the sound file for the sound that is playing

---

Sound_PreloadGroup

```
void Sound_PreloadGroup(const tString &in asInternalPath,
                        bool abSubGroups)
```

Preloads a group of sounds and its subgroups if selected Preloading removes lag spikes that can occur when loading a sound from the hdd

- **asInternalPath**: path of the sound eg. “physics/metal”
- **abSubGroups**: if the subgroups should be loaded

Sound_PreloadProject

```
void Sound_PreloadProject(const tString &in asName)
```

Preloads a whole sound project and all sounds within it Preloading removes lag spikes that can occur when loading a sound from the hdd

- **asName**: name of the sound project file

Sound_PreloadCoreSounds

```
void Sound_PreloadCoreSounds()
```

Preloads all the basic sounds

Sound_PreloadCoreUWSounds

```
void Sound_PreloadCoreUWSounds()
```

Sound_PreloadCoreInteractions

```
void Sound_PreloadCoreInteractions()
```

Sound_PreloadCoreUWInteractions

```
void Sound_PreloadCoreUWInteractions()
```

Sound_Exists

```
bool Sound_Exists(const tString &in asSoundName)
```

Returns true or false if a given sound entity exists

- **asSoundName**: name of the sound entity. Can contain wildcards.

Music_PlayExt

```
void Music_PlayExt(const tString &in asFile,
                   bool abLoop,
                   float afVolume,
                   float afFreq,
                   float afVolumeFadeTime,
                   float afFreqFadeTime,
                   eMusicPrio alPrio,
                   bool abResume)
```

Play music with extended options.

- **asFile**: name (opt: path) of the music file to play.
- **abLoop**: if the music should loop when it reaches the end.
- **afVolume**: at what volume to play the music at (0.0 - 1.0).
- **afFreq**: at what frequency to play the music, 1 = standard freq.
- **afVolumeFadeTime**: how many seconds to fade in the music, 0 = no fade.
- **afFreqFadeTime**: how many seconds to fade in the frequence change.
- **alPrio**: priority of the music track. A higher priority will stop a lower priority track.
- **abResume**: if the music should be resumed if it is stopped and played again.

Music_Play

```
void Music_Play(const tString &in asFile,
                float afVolume,
                bool abLoop,
                eMusicPrio alPrio)
```

Play music at standard frequency with a default fade time of 0.3 seconds.

- **asFile**: name (opt: path) of the music file to play.
- **afVolume**: at what volume to play the music at (0.0 - 1.0).
- **abLoop**: if the music should loop when it reaches the end.
- **alPrio**: priority of the music track. A higher priority will stop a lower priority track.

Music_PlayOverlay

```
void Music_PlayOverlay(const tString &in asFile,
                       float afVolume)
```

Plays a piece of music over the currently running music TODO: Needs to have proper code!

- **asFile**: name (opt: path) of the music file to play.
- **afVolume**: at what volume to play the music at (0.0 - 1.0).

Music_Stop

```
void Music_Stop(float afFadeTime,
                eMusicPrio alPrio)
```

Stop a music track for a certain priority from playing.

- **afFadeTime**: how many seconds it takes to fade the volume to 0.
- **alPrio**: priority of the music track that should be stopped.

Music_StopAll

```
void Music_StopAll(float afFadeTime)
```

Stop a music tracks for all priorities

- **afFadeTime**: how many seconds it takes to fade the volume to 0.

Music_AddDynamicTrack

```
void Music_AddDynamicTrack(tID a_idEntity,
                           int alTrackPrio,
                           eMusicPrio alMusicPrio,
                           const tString &in asFile,
                           float afVolume,
                           float afFadeInTime,
                           float afFadeOutTime)
```

This adds music that is meant to only be played during a certain occurance, eg when an creature is hunting the player.

- **a_idEntity**: Id of the entity that cause the occurance to happen
- **alTrackPrio**: priority of this track instnace, in case there are many. If two tracks have the same prio, music prio is compared.
- **alMusicPrio**: The prio of the music to be played.
- **asFile**: File of the music to play.
- **afVolume**: volume of the music to play
- **afFadeInTime**: time it takes for the music to fade in
- **afFadeOutTime**: time it takes for the music to fade out. Not used if there is a swithc to another dynamic track.

Music_RemoveDynamicTrack

```
void Music_RemoveDynamicTrack(tID a_idEntity)
```

This removes the dynamic track for certain entity and, if playing, fades out the music associated with it.

- **a_idEntity**: Id of the entity that cause the occurance to happen

---

Music_FadeVolumeMul

```
void Music_FadeVolumeMul(float afMul,
                         float afTime)
```

Fades the volume multiplier for music.

- **afMul**: target volume multiplier. Default is 1.
- **afTime**: time to fade to the target multiplier.

Voice_Play

```
bool Voice_Play(const tString &in asSubject,
                int alSpecificLineIdx=-1,
                const tString &in asCallback="",
                int alPrio=)
```

Starts playing a voice

- **asSubject**: The name of the subject to play.
- **alSpecificLineIdx**: if 0 or higher then a specific line index will be played. Set as -1 to play as stated in the data.
- **asCallback**: called when subject is done playing. Syntax: void Func(const tString&in asScene, const tString&in asSubject)
- **alPrio**: The priority of the voice, if higher than the currently playing, the current gets stopped and replaced, else this voice is not played.

Voice_PlayWhenPossible

```
void Voice_PlayWhenPossible(const tString &in asSubject,
                            const tString &in asConditionCallback="",
                            float afMaxCheckTime=30.0f,
                            float afMinQuietTime=5.0f,
                            const tString &in asVoiceCallback="",
                                         int alPrio=)
```

Starts playing a voice, but if another one is already playing it queues it until no other sound is playing and some custom conditions are met.

- **asSubject**: The name of the subject to play.
- **asConditionCallback**: if not “”, this callback will be called to check if the subject can be played. Syntax bool f(const tString& in asSubject)
- **afMaxCheckTime**: the max amount of time that it wil check if the conditions are met. If time runs out, the subject is never played. If below 0, then time never runs out.
- **afMinQuietTime**: the minimum amount of time there should have been no other voices playing for it is possible to play the subject. If below 0, then voice playing check is skipped.
- **asVoiceCallback**: called when subject is done playing. Syntax: void Func(const tString&in asScene, const tString&in asSubject)
- **alPrio**: The priority of the voice, if higher than the currently playing, the current gets stopped and replaced, else this voice is not played.

_Voice_PlayWhenPossible_CheckTimer

```
void _Voice_PlayWhenPossible_CheckTimer(const tString &in asTimer)
```

Voice_AbortIfQueued

```
void Voice_AbortIfQueued(const tString &in asSubject)
```

If a subject is queued (still not played) after started with Voice_PlayWhenPossible, use this to remove it and make sure it never plays

- **asSubject**: The name of the subject that is queued.

Voice_ClearQueued

```
void Voice_ClearQueued()
```

Clears all queued subjects.

Voice_IsQueued

```
bool Voice_IsQueued(const tString &in asSubject)
```

If a subject is queued with Voice_PlayWhenPossible, return true

- **asSubject**: The name of the subject that is queued.

Voice_SubjectExists

```
bool Voice_SubjectExists(const tString &in asSubject)
```

Checks if the specified subject exists.

- **asSubject**: The name of the subject

Voice_GetSubjectLineNumber

```
int Voice_GetSubjectLineNumber(const tString &in asSubject)
```

Gets the number of subject lines in a subject. Returns 0 if the subject does not exist.

- **asSubject**: The name of the subject

Voice_GetSubjectSceneName

```
tString Voice_GetSubjectSceneName(const tString &in asSubject)
```

Gets the name of the scene that the specified subject belongs to.

- **asSubject**: The name of the subject

Returns: tString , The scene the subject belongs to.

Voice_SetSource

```
void Voice_SetSource(const tString &in asCharacter,
                     const tString &in asEntityName,
                     float afMinDistance,
                     float afMaxDistance,
                     bool abUse3D,
                     float afMaxPlayerListeningRange=-1,
                     float afMinFreq=22000,
                     float afMaxFreq=22000,
                     eLuxVoiceSourceFreqencyFlag
aFrequencyFlags=eLuxVoiceSourceFreqencyFlag_None)
```

Sets the source of voice. Note that this can be a moving entity.

- **asCharacter**: The name of the character (as defined in voice file)
- **asEntityName**: The entity the sound will be attached to
- **afMinDistance**: The distance when the volume of the voice start getting lower.
- **afMaxDistance**: Max distance the voice can be heard.
- **abUse3D**: If 3D effects (panning) are used for the voice.
- **afMaxPlayerListeningRange**: The maximum range that the player is considered listening.
- **afMinFreq**: The cutoff freqeuncy at min distance 0 - 22000 hz
- **afMaxFreq**: The cutoff freqeuncy at max distance 0 - 22000 hz
- **aFrequencyFlags**: Which pass the frequency should affect. eLuxVoiceSourceFreqencyFlag, 0 = none, 1 = low pass, 2 = high pass, 3 = both

---

Voice_StopAll

```
void Voice_StopAll()
```

Stops all voices playing

Voice_Stop

```
void Voice_Stop(const tString &in asScene)
```

Stops all voices in a scene.

- **asScene**: Name of the scene (as defined in the voice file)

Voice_SkipCurrentLine

```
void Voice_SkipCurrentLine(const tString &in asScene)
```

Skips the current line (same as stop but response triggers are used)

- **asScene**: Name of the scene (as defined in the voice file)

Voice_SkipCurrentSound

```
void Voice_SkipCurrentSound(const tString &in asScene)
```

Skips the current sound and jumps to next (if any)

- **asScene**: Name of the scene (as defined in the voice file)

Voice_AdvanceFromCurrentSound

```
void Voice_AdvanceFromCurrentSound(const tString &in asScene)
```

Like SkipCurrentSound, but if there is subtitle it makes sure it is displayed first.

- **asScene**: Name of the scene (as defined in the voice file)

Voice_SetPaused

```
void Voice_SetPaused(const tString &in asScene,
                     bool abX)
```

Stops all voices in a scene.

- **asScene**: Name of the scene (as defined in the voice file)
- **abX**: If the scene is to be paused or resumed

Voice_SetPausedAll

```
void Voice_SetPausedAll(bool abX)
```

Stops all active voice scenes

- **abX**: If all active scenes is to be paused or resumed

Voice_SetFocusScene

```
void Voice_SetFocusScene(const tString &in asScene)
```

Sets the focus for a scene even if a scene with higher focusprio is playing. This makes the voices in the scene show subtitles and voices from other scenes lower. This will stay in effect until another scene is set. Note that a new subject in a scene with higher focus prio will override this.

- **asScene**: Name of the scene (as defined in the voice file). “” sets focus to no scene.

Voice_FadeSceneVolumeTo

```
void Voice_FadeSceneVolumeTo(const tString &in asScene,
                             float afVolume,
                             float afTime)
```

Fades the volume of the voices from a specfic scene

- **asScene**: Name of the scene (as defined in the voice file)
- **afVolume**: Volume to fade to,
- **afTime**: The amount of time the fade takes.

Voice_CharacterIsSpeaking

```
bool Voice_CharacterIsSpeaking(const tString &in asCharacter)
```

If a character is currently speaking (i.e their voice is played).

- **asCharacter**: Name of the character (as defined in the voice file)

Voice_SceneIsActive

```
bool Voice_SceneIsActive(const tString &in asScene)
```

If the specified scene is currently active.

- **asScene**: Name of the scene (as defined in the voice file)

Returns: bool, true if scene is currently active.

Voice_SubjectIsPlaying

```
bool Voice_SubjectIsPlaying(const tString &in asSubject)
```

If the specified subject is currently being played.

- **asSubject**: Name of the subject (as defined in the voice file)

Returns: bool, true if subject is currently active.

Voice_AnySceneIsActive

```
bool Voice_AnySceneIsActive()
```

If any scene is currently active.

Returns: bool, true if any scene is currently active.

Voice_SceneInvolvingCharacterIsActive

```
bool Voice_SceneInvolvingCharacterIsActive(const tString &in asCharacter)
```

True if a scene involving this character is currently active.

- **asCharacter**: Name of the character (as defined in the voice file)

Returns: bool, true if any scene involving the character is currently active.

Voice_SetCharacterSpeakingCallback

```
void Voice_SetCharacterSpeakingCallback(const tString &in asCharacter,
                                        const tString &in asCallback)
```

Sets a callback that is called when a character starts and stops speaking. If character callback exists, the callback func is replaced with the latest one.

- **asCharacter**: Name of the character
- **asCallback**: Callbackfunction. Syntax: bool func(const tString &in asCharacter, bool abStartedTalking), if false is returned the callback is removed.

Voice_RemoveCharacterSpeakingCallback

```
void Voice_RemoveCharacterSpeakingCallback(const tString &in asCharacter)
```

Removes a callback that is called when a character starts and stops speaking

- **asCharacter**: Name of the character

Dialog_Begin

```
void Dialog_Begin(const tString &in asName="")
```

Begins the declaration of a dialog. Must be called before any other dialog creation is done.

- **asName**: Name of the dialog

Dialog_End

```
void Dialog_End(const tString &in asStartBranch="")
```

Ends the declaration of a dialog and starts the dialog. After this no more dialog creation funcs can be called without calling Dialog_Begin again.

- **asStartBranch**: Name of the branch to start playing. If empty first declared branch is started

Dialog_SetCallbackFunc

```
void Dialog_SetCallbackFunc(const tString &in asFunc)
```

Sets a callback function for the dialog. NOTE: Currently not used!

Dialog_AddBranch

```
void Dialog_AddBranch(const tString &in asName,
                      const tString &in asNextBranch="")
```

Adds a branch (a list of subjects) to the dialog. Dialog_Begin must be called before this is used!

- **asName**: Name of the branch.
- **asNextBranch**: The branch to start playing when this has ended. If empty, no new branch is played after this is over.

Dialog_AddBranchAndSubject

```
void Dialog_AddBranchAndSubject(const tString &in asSubjectAndBranchName,
                                const tString &in asNextBranch="",
                                     const tString &in asCallback="")
```

Adds a branch with a single subject to the dialog. The branch gets the same name as the subject. Dialog_Begin must be called before this is used!

- **asSubjectAndBranchName**: Shared name of both subject and branch.
- **asNextBranch**: The branch to start playing when this has ended. If empty, no new branch is played after this is over.
- **asCallback**: Callback function run at the start and end of subject. Syntax: "void asCallback(const tString&in asSubject, bool abStartOfSubject)

Dialog_AddSubject

```
void Dialog_AddSubject(const tString &in asSubject,
                       const tString &in asCallback="")
```

Adds a subject to the latest added branch. The subjects will be played in order they are added. Dialog_AddBranch must be called before this is used!

- **asSubject**: The name of the subject. This is [scene]_[subject], e.g. AdamAndEve_HowAboutEmApples
- **asCallback**: Callback function run at the start and end of subject. Syntax: "void asCallback(const tString&in asSubject, bool abStartOfSubject)

Dialog_AddPause

```
void Dialog_AddPause(float afTime,
                     const tString &in asCallback="")
```

Adds a pause to the latest added branch. This pause will come after Dialog_AddBranch must be called before this is used!

- **afTime**: The length of the pause in seconds.
- **asCallback**: Callback function run at the start and end of pause. Syntax: "void asCallback(const tString&in asSubject, bool abStartOfSubject)

Dialog_SetResponseTimeLimit

```
void Dialog_SetResponseTimeLimit(float afTime)
```

Set the response time for the response options. This is how long the player has to pick. Dialog_AddSubject must be called before this is used!

- **afTime**: The time before the selection ends. 0=forever, this is default if not set.

Dialog_AddResponseOption

```
void Dialog_AddResponseOption(const tString &in asEntry,
                              const tString &in asBranch,
                              int alOptionId=-1,
                              const tString &in asCallback="")
```

Adds a response option that is shown after the subject is over. Max number is 4, +1 default (this actually depends on the game implementation, but this the default max). Important: The defauly subject must be added last! Dialog_AddSubject must be called before this is used!

- **asEntry**: This is the entry in the lang file, the category is the level name. If ”” this option is used if time runs out. If needed, it must be added last!
- **asBranch**: Branch that will be switched to if chosen. If ”“, no new branch is played after this is over.
- **alOptionId**: Used in callback to see which option was picked (index is not good to use, becuase it is not always the same)
- **asCallback**: Called when the option has been chosen. Syntax: void asCallback(const tString&in asBranch, const tString&in asBranchSubject, int alOptionId)

Dialog_AddResponseCondition_VarIsSet

```
void Dialog_AddResponseCondition_VarIsSet(const tString &in asVar)
```

Adds a condition that must be true for the option reponse option to be added. This checks if var is greater than 0. Dialog_AddResponseOption must be called before this is used!

- **asVar**: Name of the variable.

Dialog_AddResponseCondition_VarNotSet

```
void Dialog_AddResponseCondition_VarNotSet(const tString &in asVar)
```

Adds a condition that must be true for the option reponse option to be added. This checks if var is less than 1 Dialog_AddResponseOption must be called before this is used!

- **asVar**: Name of the variable.

Dialog_AddResponseCondition_VarEqual

```
void Dialog_AddResponseCondition_VarEqual(const tString &in asVar,
                                          int alVal)
```

Adds a condition that must be true for the option reponse option to be added. This checks if var is equal to a value. Dialog_AddResponseOption must be called before this is used!

- **asVar**: Name of the variable.
- **alVal**: Value check is made against.

Dialog_AddResponseCondition_VarLesser

```
void Dialog_AddResponseCondition_VarLesser(const tString &in asVar,
                                           int alVal)
```

Adds a condition that must be true for the option reponse option to be added. This checks if var is lesser than a value. Dialog_AddResponseOption must be called before this is used!

- **asVar**: Name of the variable.
- **alVal**: Value check is made against.

Dialog_AddResponseCondition_VarGreater

```
void Dialog_AddResponseCondition_VarGreater(const tString &in asVar,
                                            int alVal)
```

Adds a condition that must be true for the option reponse option to be added. This checks if var is greater than a value. Dialog_AddResponseOption must be called before this is used!

- **asVar**: Name of the variable.
- **alVal**: Value check is made against.

Dialog_AddResponseEvent_SetVar

```
void Dialog_AddResponseEvent_SetVar(const tString &in asVar,
                                    int alVal=1)
```

Event that happens after option is picked. This sets the value of a variable. Dialog_AddResponseOption must be called before this is used!

- **asVar**: Name of the variable.

- **alVal**: Value that is set

Dialog_AddResponseEvent_IncVar

```
void Dialog_AddResponseEvent_IncVar(const tString &in asVar,
                                    int alVal=1)
```

Event that happens after option is picked. This increments the value of a variable. Dialog_AddResponseOption must be called before this is used!

- **asVar**: Name of the variable.
- **alVal**: Value that is added to the variabel.

Dialog_AddResponse_OneTimeCheck

```
void Dialog_AddResponse_OneTimeCheck(const tString &in asVar)
```

This makes sure that the option is only shown once. Dialog_AddResponseOption must be called before this is used!

- **asVar**: Name of the variable to keep track of this.

Dialog_AddEndEvent_SetVar

```
void Dialog_AddEndEvent_SetVar(const tString &in asVar,
                               int alValue=1)
```

Sets the value of an internal Dialog variableafter a subject is over. Dialog_AddSubject must be called before this is used!

- **asVar**: The name of the var
- **alValue**: The value to set the var.

Dialog_AddEndEvent_IncVar

```
void Dialog_AddEndEvent_IncVar(const tString &in asVar,
                               int alValue=1)
```

Increments the value of an internal Dialog variable after a subject is over. If var was not set priorly, it will start with 0. Dialog_AddSubject must be called before this is used!

- **asVar**: The name of the var
- **alValue**: The value to set the var.

Dialog_AddEndEvent_VarIsSet

```
void Dialog_AddEndEvent_VarIsSet(const tString &in asVar,
                                 const tString &in asNewBranch)
```

Checks if a variable as a value greater than 0 and exists. Dialog_AddSubject must be called before this is used!

- **asVar**: The name of the var
- **asNewBranch**: The name of the dialog to branch to. If empty, the dialog just ends.

Dialog_AddEndEvent_VarEquals

```
void Dialog_AddEndEvent_VarEquals(const tString &in asVar,
                                  int alValue,
                                  const tString &in asNewBranch)
```

Checks if a variable is equal to a value. If a variable does not exist it will assume to have value 0. Dialog_AddSubject must be called before this is used!

- **asVar**: The name of the var
- **alValue**: value the variable is checked against
- **asNewBranch**: The name of the dialog to branch to. If empty, the dialog just ends.

Dialog_AddEndEvent_VarGreater

```
void Dialog_AddEndEvent_VarGreater(const tString &in asVar,
                                   int alValue,
                                   const tString &in asNewBranch)
```

Checks if a variable is greater than a value. If a variable does not exist it will assume to have value 0. Dialog_AddSubject must be called before this is used!

- **asVar**: The name of the var
- **alValue**: value the variable is checked against
- **asNewBranch**: The name of the dialog to branch to. If empty, the dialog just ends.

Dialog_AddEndEvent_VarLesser

```
void Dialog_AddEndEvent_VarLesser(const tString &in asVar,
                                  int alValue,
                                  const tString &in asNewBranch)
```

Checks if a variable is lesser than a value. If a variable does not exist it will assume to have value 0. Dialog_AddSubject must be called before this is used!

- **asVar**: The name of the var
- **alValue**: value the variable is checked against
- **asNewBranch**: The name of the dialog to branch to. If empty, the dialog just ends.

Dialog_AddLineEvent_OutOfRange

```
void Dialog_AddLineEvent_OutOfRange(const tString &in asNewBranch)
```

Adds a branching event to the latest added subject. Checked at end of every line. This event checks if player is out of range from all characters in the dialog, and if so branches. Dialog_AddSubject must be called before this is used!

- **asNewBranch**: The name of the dialog to branch to. If empty, the dialog just ends.

Dialog_AddEndEvent_OutOfRange

```
void Dialog_AddEndEvent_OutOfRange(const tString &in asNewBranch)
```

Adds a branching event to the latest added subject. Checked at end of the subject. This event checks if player is out of range from all characters in the dialog, and if so branches. Dialog_AddSubject must be called before this is used!

- **asNewBranch**: The name of the dialog to branch to. If empty, the dialog just ends.

Dialog_AddLineEvent_PlayerNotLooking

```
void Dialog_AddLineEvent_PlayerNotLooking(const tString &in asNewBranch)
```

Adds a branching event to the latest added subject. Checked at end of every line. This event checks if is not looking at any character in the dialog, and if so branches. Dialog_AddSubject must be called before this is used!

- **asNewBranch**: The name of the dialog to branch to. If empty, the dialog just ends.

Dialog_AddEndEvent_PlayerNotLooking

```
void Dialog_AddEndEvent_PlayerNotLooking(const tString &in asNewBranch)
```

Adds a branching event to the latest added subject. Checked at end of the subject. This event checks if is not looking at any character in the dialog, and if so branches. Dialog_AddSubject must be called before this is used!

- **asNewBranch**: The name of the dialog to branch to. If empty, the dialog just ends.

Dialog_AddLineEvent_PlayerNotLookingOrOutOfRange

```
void Dialog_AddLineEvent_PlayerNotLookingOrOutOfRange(const tString &in
asNewBranch)
```

Adds a branching event to the latest added subject. Checked at end of every line. This event checks if is not looking at any character any character or is out of range, and if so branches. Dialog_AddSubject must be called before this is used!

- **asNewBranch**: The name of the dialog to branch to. If empty, the dialog just ends.

Dialog_AddEndEvent_PlayerNotLookingOrOutOfRange

```
void Dialog_AddEndEvent_PlayerNotLookingOrOutOfRange(const tString &in
asNewBranch)
```

Adds a branching event to the latest added subject. Checked at end of subject. This event checks if is not looking at any character any character or is out of range, and if so branches. Dialog_AddSubject must be called before this is used!

- **asNewBranch**: The name of the dialog to branch to. If empty, the dialog just ends.

Dialog_AddLineEvent_Callback

```
void Dialog_AddLineEvent_Callback(const tString &in asCallbackFunc,
                                  const tString &in asNewBranch)
```

Adds a branching event to the latest added subject. Checked at end of every line. This event checks if a custom callback returns true and if so branches. Dialog_AddSubject must be called before this is used!

- **asCallbackFunc**: The callback function name. Syntax: bool asCallbackFunc(const tString&in asBranch, const tString&in asBranchSubject, int alLineIndex, const tString&in asNewBranch)”
- **asNewBranch**: The name of the dialog to branch to. If empty, the dialog just ends.

Dialog_AddEndEvent_Callback

```
void Dialog_AddEndEvent_Callback(const tString &in asCallbackFunc,
                                 const tString &in asNewBranch)
```

Adds a branching event to the latest added subject. Checked at end of the subject. This event checks if a custom callback returns true and if so branches. Dialog_AddSubject must be called before this is used!

- **asCallbackFunc**: The callback function name. Syntax: “bool asCallbackFunc(const tString&in asBranch, const tString&in asBranchSubject, int alLineIndex, const tString&in asNewBranch)”
- **asNewBranch**: The name of the dialog to branch to. If empty, the dialog just ends.

Dialog_CharacterIsActive

```
bool Dialog_CharacterIsActive(const tString &in asName)
```

Checks if a character is currently taking part in a dialog.

- **asName**: Name of the character.

Dialog_Stop

```
void Dialog_Stop(const tString &in asName)
```

Stops a dialog

- **asName**: Name of the dialog

Dialog_StopAll

```
void Dialog_StopAll()
```

Stops all dialog

Dialog_StartConversation_CharMover

```
void Dialog_StartConversation_CharMover(const tString &in asVoiceCharacter,
                                        const tString &in asFocusEntityName,
                                        int alHeadBoneIndex,
                                        cLuxCharMover @apCharMover,
                                        float afMinTurnAngle,
                                        float afMaxMoveDist=0.5f)
```

Starts a dialog with an entity that has a CharMover. Should only be used internally by entities.

Dialog_StartConversation_PosBased

```
void Dialog_StartConversation_PosBased(const tString &in asVoiceCharacter,
                                       const tString &in asFocusEntity,
                                       const tString &in asPositionArea,
                                       const tString &in asHeadBone="",
                                       float afMaxMoveDist=0.5f,
                                       cVector3f
avFocusOffset=cVector3f_Zero)
```

Starts a dialog with an entity, moving the player to a position to have the conversation.

- **asVoiceCharacter**: the character with wich the player will have the dialog.
- **asFocusEntity**: the entity the player will look at during the dialog.
- **asPositionArea**: the area the player will be moved to at the start of the conversation (centre base of area = foot position)
- **asHeadBone**: if non-“”, the bone of asFocusEntity that should be focused on during the conversation (e.g. “head”)
- **afMaxMoveDist**: the max distance that can be moved from where the player is standing when the dialog starts.
- **avFocusOffset**: any offset that should be added to the focus position (origin of focusEntity, or head bone position if specified)

Dialog_StartConversation

```
void Dialog_StartConversation(const tString &in asVoiceCharacter,
                              const tString &in asFocusEntityName,
                              float afMaxMoveDist=0.5f,
                              cVector3f avFocusOffset=cVector3f_Zero)
```

Starts a dialog with an entity.

- **asVoiceCharacter**: the character with wich the player will have the dialog.
- **asFocusEntityName**: the entity the player will look at during the dialog.
- **afMaxMoveDist**: the max distance that can be moved from where the player is standing when the dialog starts.
- **avFocusOffset**: any offset that should be added to the focus position (origin of focusEntity, or head bone position if specified)

Dialog_GetCharacterScene

```
tString Dialog_GetCharacterScene(const tString &in asCharacter)
```

Gets the scene the character is currently in.

- **asCharacter**: Name of the character (as defined in the voice file)

Dialog_SetVar

```
void Dialog_SetVar(const tString &in asVar,
                   int alValue=1)
```

Sets the value of an internal Dialog variable

- **asVar**: The name of the var
- **alValue**: The value to set the var to.

Dialog_IncVar

```
void Dialog_IncVar(const tString &in asVar,
                   int alValue=1)
```

Incremements the value of an internal Dialog variable. If var was not set priorly, it will start with 0.

- **asVar**: The name of the var
- **alValue**: The value to add to the var.

Dialog_GetVar

```
int Dialog_GetVar(const tString &in asVar)
```

Gets the value of an internal Dialog variable. If it does not exist, 0 is returned.

- **asVar**: The name of the var
- **alValue**: The value to set the var.

Dialog_GetCharactersInSubject

```
void Dialog_GetCharactersInSubject(const tString &in asSubject,
                                   array< tString > &out avOutCharacters)
```

Returns a list of the characters who speak in a particular dialog subject.

- **asSubject**: name of subject
- **avOutCharacters**: array<tString> to hold the resulting list

---