ThereMelo

Generated by Doxygen 1.10.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

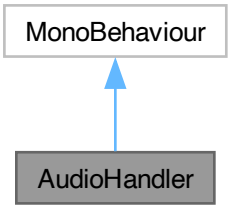
Chapter 3

Class Documentation

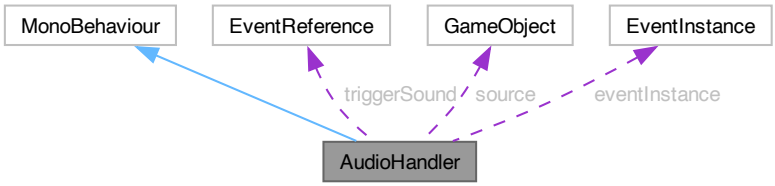
3.1 AudioHandler Class Reference

Handles audio playback and effects for a virtual music instrument in ThereMelo.

Inheritance diagram for AudioHandler:



Collaboration diagram for AudioHandler:



Public Member Functions

- void PlayAudioOnce ()

  Plays the trigger sound once at the source's current position.
- void PlayAudioConstant ()

  Starts playing the trigger sound continuously.
- void PlayForAmount (float setTime)

  Plays the trigger sound continuously for a specified amount of time.
- void StopAudio ()

  Stops the currently playing audio with a fade out.

Private Member Functions

- void Awake ()

  Initializes the audio source to the main camera if not set.
- IEnumerator StopSoundAfterDelay (float delay)

  Coroutine to stop sound after a specified delay.

Private Attributes

- EventReference triggerSound

  Reference to the FMOD event for the trigger sound.
- GameObject source

  GameObject that acts as the source of the sound.
- EventInstance eventInstance

  Instance of an FMOD event.

3.1.1 Detailed Description

Handles audio playback and effects for a virtual music instrument in ThereMelo.

This class is used to manipulate audio wheher it be to play or stop playing audio. This class is used by others inorder to trigger audio feedback. See the diagram for full details on these interations

3.1.2 Member Function Documentation

3.1.2.1 Awake()

```
void AudioHandler.Awake ( )  [inline], [private]
```

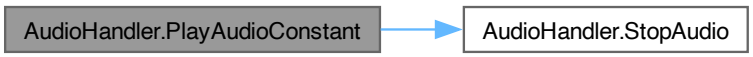
Initializes the audio source to the main camera if not set.

3.1.2.2 PlayAudioConstant()

```
void AudioHandler.PlayAudioConstant ( )  [inline]
```

Starts playing the trigger sound continuously.

Stops the current audio if it is already playing before starting the new audio playback. Here is the call graph for this function:



3.1.2.3 PlayAudioOnce()

```
void AudioHandler.PlayAudioOnce ( )  [inline]
```

Plays the trigger sound once at the source's current position.

3.1.2.4 PlayForAmount()

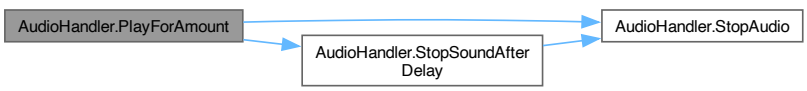
```
void AudioHandler.PlayForAmount (
            float setTime )  [inline]
```

Plays the trigger sound continuously for a specified amount of time.

Parameters

setTime	The duration in seconds to play the sound for.

Here is the call graph for this function:

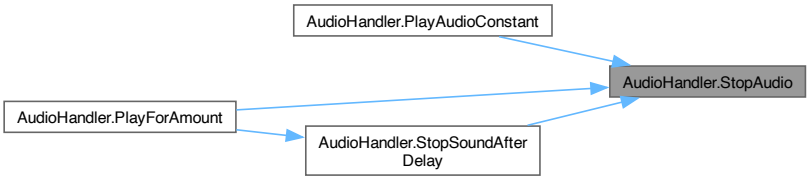


3.1.2.5 StopAudio()

```
void AudioHandler.StopAudio ( )  [inline]
```

Stops the currently playing audio with a fade out.

Here is the caller graph for this function:



3.1.2.6 StopSoundAfterDelay()

```
IEnumerator AudioHandler.StopSoundAfterDelay (
            float delay )  [inline], [private]
```

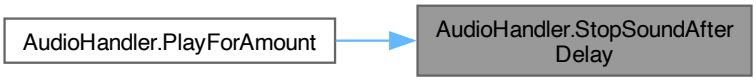
Coroutine to stop sound after a specified delay.

Parameters

delay	The delay in seconds before stopping the sound.

Here is the call graph for this function:

Here is the caller graph for this function:



3.1.3 Member Data Documentation

3.1.3.1 eventInstance

```
EventInstance AudioHandler.eventInstance  [private]
```

Instance of an FMOD event.

3.1.3.2 source

```
GameObject AudioHandler.source  [private]
```

GameObject that acts as the source of the sound.

3.1.3.3 triggerSound

```
EventReference AudioHandler.triggerSound  [private]
```

Reference to the FMOD event for the trigger sound.

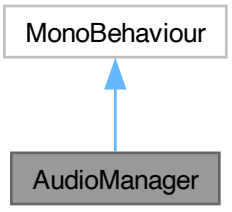
The documentation for this class was generated from the following file:

- Audio/AudioHandler.cs

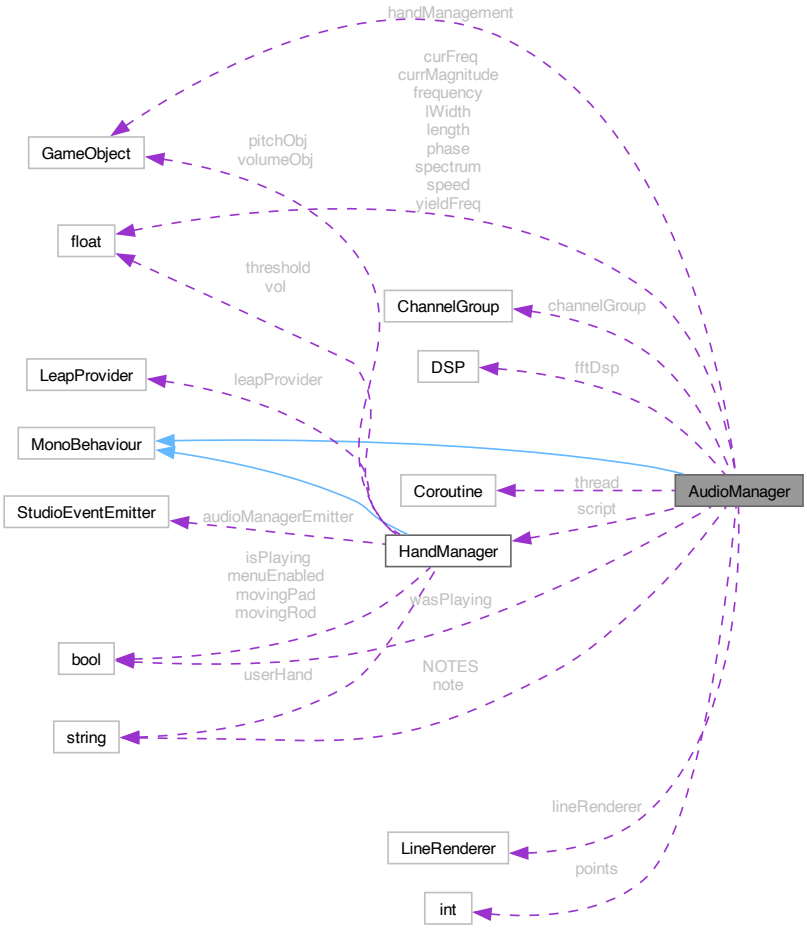
3.2 AudioManager Class Reference

This class is used to manage calculate and.

Inheritance diagram for AudioManager:



Collaboration diagram for AudioManager:



Public Member Functions

- float **getMagnitude** ()
- float **getFreq** ()
- void PlayOneShot (EventReference sound, Vector3 worldPos)

  Plays an audio clip once at a specified position in the world.
- EventInstance PlayConstaint (EventReference sound, Vector3 worldPos)

  Creates and plays an audio event continuously at a specified position in the world.

Public Attributes

- GameObject **handManagement**

Properties

- static AudioManager instance `[get, private set]`

  Single instance of AudioManager to manage audio across the scene.

Private Member Functions

- void Awake ()

  Initializes the AudioManager instance and sets up audio analysis tools.
- void Update ()

  Updates audio analysis and visualizer based on current audio playback state.
- IEnumerator FFTAnalysisCoroutine ()

  Coroutine to perform FFT analysis at set intervals.
- UnityEngine.Color ConvertColor ((int, int, int) colorTuple)

  Converts a color tuple to a UnityEngine.Color.
- string getNote (float freq)

  Determines the musical note corresponding to a given frequency.
- void FFTAnalysis ()

  Performs FFT analysis to detect frequency and amplitude of the current audio.

Private Attributes

- ChannelGroup channelGroup

  Group of channels for managing audio playback.
- DSP fftDsp

  Digital Signal Processing (DSP) unit for performing Fast Fourier Transform (FFT) analysis.
- float **yieldFreq** = 3f
- Coroutine **thread** = null
- bool wasPlaying = false

  Flag to track if audio was playing in the previous frame.
- HandManager **script**
- float **currMagnitude** = 0f
- float[ ] **spectrum** = new float[512]
- float **frequency**
- float **curFreq**
- string **note**
- LineRenderer lineRenderer

  Renderer to visualize audio data.
- int **points** = 25
- float **length** = 5f
- float **lWidth** = 0.025f
- float **phase**
- float **speed** = 1f
- string[ ] **NOTES** = { "A", "A#", "B", "C", "C#", "D", "D#", "E", "F", "F#", "G", "G#" }

3.2.1 Detailed Description

This class is used to manage calculate and.

3.2.2 Member Function Documentation

3.2.2.1 Awake()

```
void AudioManager.Awake ( )  [inline], [private]
```

Initializes the AudioManager instance and sets up audio analysis tools.

3.2.2.2 ConvertColor()

```
UnityEngine.Color AudioManager.ConvertColor (
            (int, int, int) colorTuple )  [inline], [private]
```

Converts a color tuple to a UnityEngine.Color.

Parameters

colorTuple	Tuple representing the RGB values of the color.

Returns

Converted UnityEngine.Color.

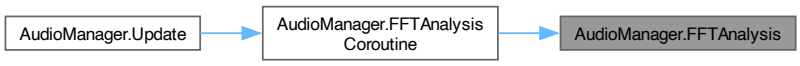
3.2.2.3 FFTAnalysis()

```
void AudioManager.FFTAnalysis ( )  [inline], [private]
```

Performs FFT analysis to detect frequency and amplitude of the current audio.

Here is the call graph for this function:

Here is the caller graph for this function:



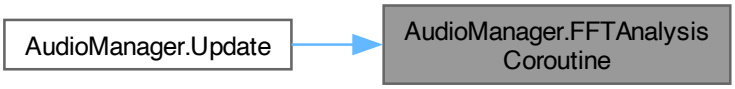
3.2.2.4 FFTAnalysisCoroutine()

```
IEnumerator AudioManager.FFTAnalysisCoroutine ( )  [inline], [private]
```

Coroutine to perform FFT analysis at set intervals.

Here is the call graph for this function:

Here is the caller graph for this function:



3.2.2.5 getNote()

```
string AudioManager.getNote (
            float freq )  [inline], [private]
```

Determines the musical note corresponding to a given frequency.

Parameters

freq	Frequency to determine the note for.

Returns

String representing the musical note.

Here is the caller graph for this function:

3.2.2.6 PlayConstaint()

```
EventInstance AudioManager.PlayConstaint (
            EventReference sound,
            Vector3 worldPos )  [inline]
```

Creates and plays an audio event continuously at a specified position in the world.

Parameters

sound	Reference to the FMOD event to play.
worldPos	World position to play the sound at.

Returns

The EventInstance of the playing sound for further manipulation.

3.2.2.7 PlayOneShot()

```
void AudioManager.PlayOneShot (
            EventReference sound,
            Vector3 worldPos )  [inline]
```

Plays an audio clip once at a specified position in the world.

Parameters

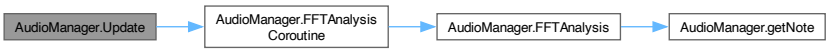
sound	Reference to the FMOD event to play.
worldPos	World position to play the sound at.

3.2.2.8 Update()

```
void AudioManager.Update ( )  [inline], [private]
```

Updates audio analysis and visualizer based on current audio playback state.

Here is the call graph for this function:



3.2.3 Member Data Documentation

3.2.3.1 channelGroup

```
ChannelGroup AudioManager.channelGroup  [private]
```

Group of channels for managing audio playback.

3.2.3.2 fftDsp

```
DSP AudioManager.fftDsp  [private]
```

Digital Signal Processing (DSP) unit for performing Fast Fourier Transform (FFT) analysis.

3.2.3.3 lineRenderer

```
LineRenderer AudioManager.lineRenderer  [private]
```

Renderer to visualize audio data.

3.2.3.4 wasPlaying

```
bool AudioManager.wasPlaying = false  [private]
```

Flag to track if audio was playing in the previous frame.

3.2.4 Property Documentation

3.2.4.1 instance

```
AudioManager AudioManager.instance  [static], [get], [private set]
```

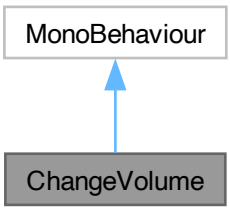
Single instance of AudioManager to manage audio across the scene.

The documentation for this class was generated from the following file:

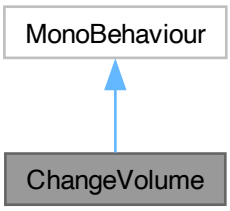
- Audio/AudioManager.cs

3.3 ChangeVolume Class Reference

Inheritance diagram for ChangeVolume:



Collaboration diagram for ChangeVolume:



Public Member Functions

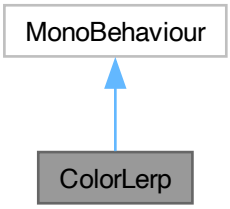
- void **ChangeMasterVolValue** ()

The documentation for this class was generated from the following file:

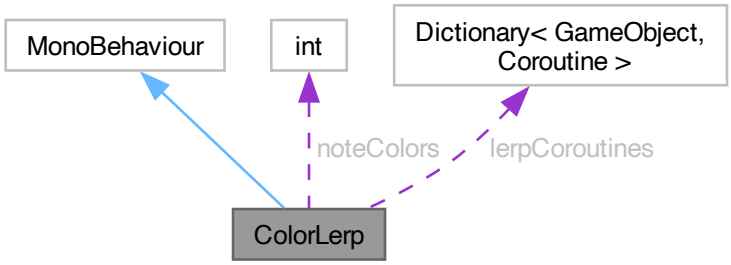
- Other/ChangeVolume.cs

3.4 ColorLerp Class Reference

Inheritance diagram for ColorLerp:



Collaboration diagram for ColorLerp:



Public Member Functions

- void **startLerp** (int noteIndex)

Private Member Functions

- Color **convertColor** ((int, int, int) colorTuple)
- IEnumerator **lerpColor** (GameObject obj, Color targetColor, float dur)

Private Attributes

- **int**
- int[ ] noteColors
- Dictionary< GameObject, Coroutine > **lerpCoroutines** = new Dictionary<GameObject, Coroutine>()

3.4.1 Member Data Documentation

3.4.1.1 noteColors

```
int [ ] ColorLerp.noteColors  [private]
```

Initial value:

```
= {
        (58, 92, 255),
        (103, 58, 255),
        (205, 58, 255),
        (255, 76, 58),
        (255, 133, 58),
        (255, 153, 58),
        (255, 215, 58),
        (162, 255, 58),
        (100, 255, 58),
        (67, 255, 58),
        (58, 255, 180),
        (58, 167, 255),
    }
```

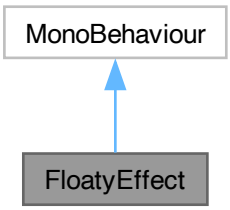
The documentation for this class was generated from the following file:

- Other/ColorLerp.cs

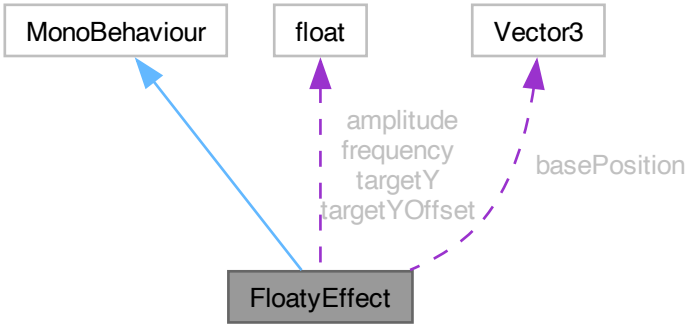
3.5 FloatyEffect Class Reference

Creates a floating effect on the GameObject to which it's attached by periodically adjusting its vertical position.

Inheritance diagram for FloatyEffect:



Collaboration diagram for FloatyEffect:



Public Attributes

- float amplitude = 0.5f

  Amplitude of the floating effect, determining the maximum height the GameObject moves up and down.
- float frequency = 1f

  Frequency of the floating effect, determining how fast the GameObject moves up and down.
- float targetYOffset = 1f

  The vertical offset from the starting position to the target position.

Private Member Functions

- void Start ()

  Initializes the effect by recording the base position and calculating the target vertical position.
- void Update ()

  Updates the vertical position of the GameObject to create a floating effect.

Private Attributes

- Vector3 basePosition

  The base position of the GameObject, set at Start.
- float targetY

  The target vertical position of the GameObject.

3.5.1 Detailed Description

Creates a floating effect on the GameObject to which it's attached by periodically adjusting its vertical position.

3.5.2 Member Function Documentation

3.5.2.1 Start()

```
void FloatyEffect.Start ( )  [inline], [private]
```

Initializes the effect by recording the base position and calculating the target vertical position.

3.5.2.2 Update()

```
void FloatyEffect.Update ( )  [inline], [private]
```

Updates the vertical position of the GameObject to create a floating effect.

3.5.3 Member Data Documentation

3.5.3.1 amplitude

```
float FloatyEffect.amplitude = 0.5f
```

Amplitude of the floating effect, determining the maximum height the GameObject moves up and down.

3.5.3.2 basePosition

```
Vector3 FloatyEffect.basePosition  [private]
```

The base position of the GameObject, set at Start.

3.5.3.3 frequency

```
float FloatyEffect.frequency = 1f
```

Frequency of the floating effect, determining how fast the GameObject moves up and down.

3.5.3.4 targetY

```
float FloatyEffect.targetY  [private]
```

The target vertical position of the GameObject.

3.5.3.5 targetYOffset

```
float FloatyEffect.targetYOffset = 1f
```

The vertical offset from the starting position to the target position.

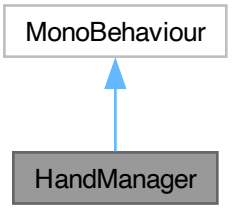
The documentation for this class was generated from the following file:

- Other/FloatyEffect.cs

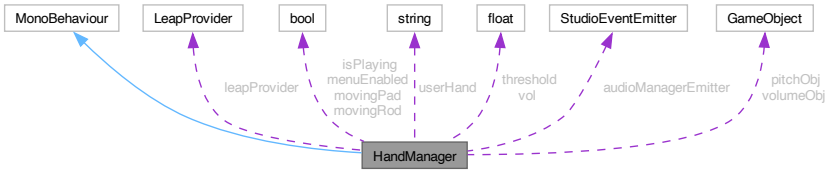
3.6 HandManager Class Reference

Manages hand interactions for ThereMelo, handling input from Leap Motion to control audio parameters.

Inheritance diagram for HandManager:



Collaboration diagram for HandManager:



Public Member Functions

- float **getVolume** ()
- void **setMenuBoolean** (bool x)
- void **setMovePad** (bool x)
- void **setMoveRod** (bool x)

Public Attributes

- LeapProvider **leapProvider**
- bool movingPad = false

  Flags to indicate the movement state of the pad, rod, and whether the menu is enabled.
- bool **movingRod** = false
- bool **menuEnabled** = false
- bool **isPlaying**

Properties

- bool **movingObject** `[get]`
- static HandManager **instance** `[get, private set]`

Private Member Functions

- void Awake ()

  Ensures a single instance of HandManager and initializes it.
- void updateHand (string key, object value)

  Updates the user's hand preference.
- void OnEnable ()

  Subscribes to Leap Motion frame updates and user preference changes.
- void OnDisable ()

  Unsubscribes from Leap Motion frame updates and user preference changes when disabled.
- bool IsPlaying (EventInstance instance)

  Checks if the audio is currently playing.
- float calcClosest (Hand hand, GameObject targetObj)

  Calculates the closest distance from the hand to a target object.
- void OnUpdateFrame (Frame frame)

  Processes hand data each frame to control audio playback and parameters based on hand position and movement.

Private Attributes

- string userHand

  Stores the chirality of the user's hand preferred for interaction.
- float **vol**
- float **threshold** = 0.05f
- StudioEventEmitter **audioManagerEmitter**
- GameObject volumeObj

  Objects representing the control points for volume and pitch.
- GameObject **pitchObj**

3.6.1 Detailed Description

Manages hand interactions for ThereMelo, handling input from Leap Motion to control audio parameters.

3.6.2 Member Function Documentation

3.6.2.1 Awake()

```
void HandManager.Awake ( )  [inline], [private]
```

Ensures a single instance of HandManager and initializes it.

3.6.2.2 calcClosest()

```
float HandManager.calcClosest (
            Hand hand,
            GameObject targetObj )  [inline], [private]
```

Calculates the closest distance from the hand to a target object.

Parameters

hand	The hand to calculate the distance from.
targetObj	The target object.

Returns

The closest distance between the hand and the target object.

Here is the caller graph for this function:

3.6.2.3 IsPlaying()

```
bool HandManager.IsPlaying (
            EventInstance instance )  [inline], [private]
```

Checks if the audio is currently playing.

Parameters

instance	The FMOD event instance to check.

Returns

True if the event instance is playing; false otherwise.

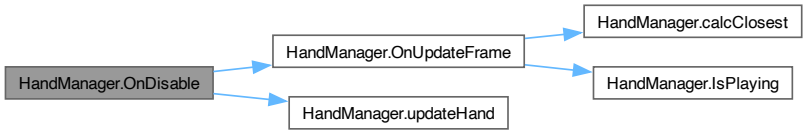
Here is the caller graph for this function:

3.6.2.4 OnDisable()

```
void HandManager.OnDisable ( )  [inline], [private]
```

Unsubscribes from Leap Motion frame updates and user preference changes when disabled.

Here is the call graph for this function:

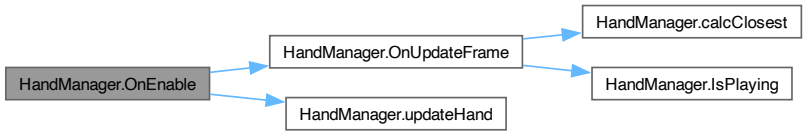


3.6.2.5 OnEnable()

```
void HandManager.OnEnable ( )  [inline], [private]
```

Subscribes to Leap Motion frame updates and user preference changes.

Here is the call graph for this function:



3.6.2.6 OnUpdateFrame()

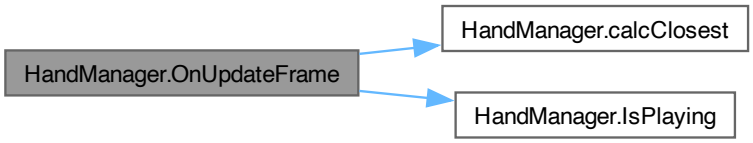
```
void HandManager.OnUpdateFrame (
            Frame frame )  [inline], [private]
```

Processes hand data each frame to control audio playback and parameters based on hand position and movement.

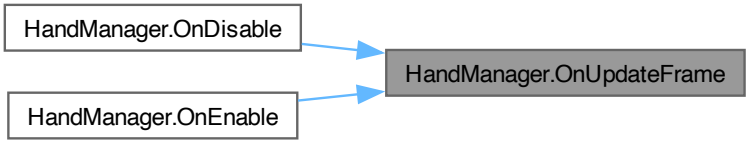
Parameters

frame	The current frame of hand data from Leap Motion.

Here is the call graph for this function:



Here is the caller graph for this function:



3.6.2.7 updateHand()

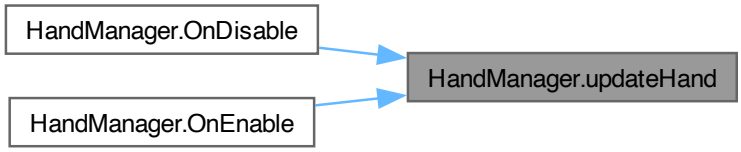
```
void HandManager.updateHand (
            string key,
            object value )  [inline], [private]
```

Updates the user's hand preference.

Parameters

key	The preference key to be updated.
value	The new value for the preference.

Here is the caller graph for this function:



3.6.3 Member Data Documentation

3.6.3.1 movingPad

```
bool HandManager.movingPad = false
```

Flags to indicate the movement state of the pad, rod, and whether the menu is enabled.

3.6.3.2 userHand

```
string HandManager.userHand  [private]
```

Stores the chirality of the user's hand preferred for interaction.

3.6.3.3 volumeObj

```
GameObject HandManager.volumeObj  [private]
```

Objects representing the control points for volume and pitch.

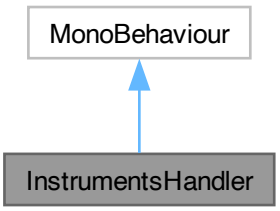
The documentation for this class was generated from the following file:

- Hands/HandManager.cs

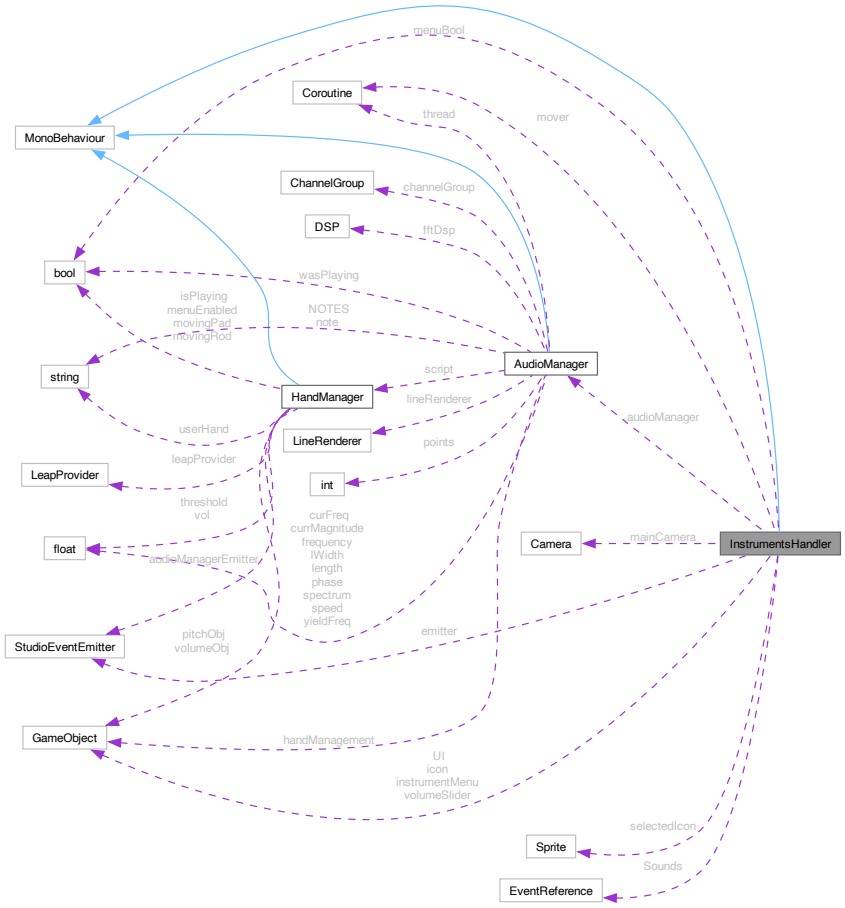
3.7 InstrumentsHandler Class Reference

Handles instrument sound changes and UI interactions for selecting different instruments.

Inheritance diagram for InstrumentsHandler:



Collaboration diagram for InstrumentsHandler:



Public Member Functions

- void changeSound (string soundName)

  Changes the sound of the instrument to the one specified by soundName.
- void toggleMenu (bool forceClose)

  Toggles the instrument menu open or closed, optionally forcing it to close.

Private Member Functions

- void Start ()

  Initializes the handler, finding necessary components in the UI.
- IEnumerator MoveTo (Vector3 target, float dur)

  Coroutine that moves the volume slider to the target position over the specified duration.

Private Attributes

- bool menuBool = false

  Tracks the state of the menu (open or closed).
- Coroutine **mover**
- GameObject volumeSlider

  References to the volume slider, instrument menu, and icon within the UI.
- GameObject **instrumentMenu**
- GameObject **icon**
- Camera **mainCamera**
- AudioManager audioManager

  Reference to the AudioManager for playing sounds.
- GameObject UI

  The UI GameObject that contains the volume slider and instrument menu.
- Sprite selectedIcon

  The icon to display when an instrument is selected.
- EventReference[ ] Sounds

  An array of EventReferences representing the different sounds for each instrument.
- StudioEventEmitter emitter

  StudioEventEmitter used to emit sound events.

3.7.1 Detailed Description

Handles instrument sound changes and UI interactions for selecting different instruments.

3.7.2 Member Function Documentation

3.7.2.1 changeSound()

```
void InstrumentsHandler.changeSound (
            string soundName )  [inline]
```

Changes the sound of the instrument to the one specified by soundName.

Parameters

soundName	The name of the sound to change to.

3.7.2.2 MoveTo()

```
IEnumerator InstrumentsHandler.MoveTo (
            Vector3 target,
            float dur )  [inline], [private]
```

Coroutine that moves the volume slider to the target position over the specified duration.

Parameters

target	The target position for the volume slider.
dur	The duration over which to move the slider.

Returns

IEnumerator for coroutine management.

Here is the caller graph for this function:

3.7.2.3 Start()

```
void InstrumentsHandler.Start ( )  [inline], [private]
```

Initializes the handler, finding necessary components in the UI.

3.7.2.4 toggleMenu()

```
void InstrumentsHandler.toggleMenu (
            bool forceClose )  [inline]
```

Toggles the instrument menu open or closed, optionally forcing it to close.

Parameters

forceClose	If true, forces the menu to close; otherwise, toggles the menu based on its current state.

Here is the call graph for this function:

3.7.3 Member Data Documentation

3.7.3.1 audioManager

```
AudioManager InstrumentsHandler.audioManager  [private]
```

Reference to the AudioManager for playing sounds.

3.7.3.2 emitter

```
StudioEventEmitter InstrumentsHandler.emitter  [private]
```

StudioEventEmitter used to emit sound events.

3.7.3.3 menuBool

```
bool InstrumentsHandler.menuBool = false  [private]
```

Tracks the state of the menu (open or closed).

3.7.3.4 selectedIcon

```
Sprite InstrumentsHandler.selectedIcon  [private]
```

The icon to display when an instrument is selected.

3.7.3.5 Sounds

```
EventReference [] InstrumentsHandler.Sounds  [private]
```

An array of EventReferences representing the different sounds for each instrument.

3.7.3.6 UI

```
GameObject InstrumentsHandler.UI  [private]
```

The UI GameObject that contains the volume slider and instrument menu.

3.7.3.7 volumeSlider

```
GameObject InstrumentsHandler.volumeSlider  [private]
```

References to the volume slider, instrument menu, and icon within the UI.

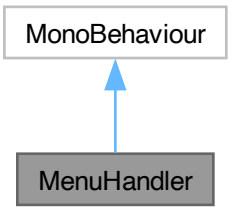
The documentation for this class was generated from the following file:

- Menu/InstrumentsHandler.cs

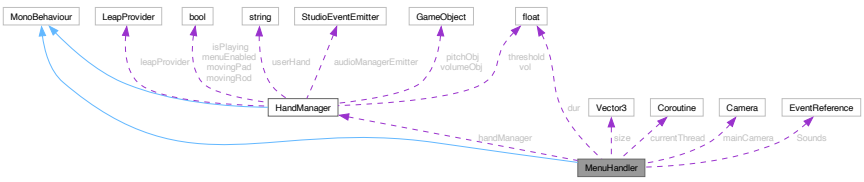
3.8 MenuHandler Class Reference

Manages the appearance and disappearance of a menu through animation and audio feedback.

Inheritance diagram for MenuHandler:



Collaboration diagram for MenuHandler:



Public Member Functions

- void toggleMenu (bool forceClose)

  Toggles the menu's visibility, optionally forcing it to close, with animations and sound effects.

Private Member Functions

- void Start ()

  Initializes the handler, finding the main camera.
- float easeInSine (float x)

  Easing function for the animation, creating a smooth start effect.
- IEnumerator Grow ()

  Coroutine to animate the menu growing to its target size.
- IEnumerator Shrink ()

  Coroutine to animate the menu shrinking to disappear.

Private Attributes

- Vector3 **size** = new Vector3(0.7f, 0.4f, 0.0001f)
- Coroutine currentThread

  Coroutine reference for the currently running animation (growing or shrinking the menu).
- float dur = 0.075f

  Duration of the animation for opening or closing the menu.
- Camera **mainCamera**
- HandManager handManager

  Reference to the HandManager to check and set the menu's enabled state.
- EventReference[ ] Sounds

  An array of EventReferences for playing sound effects associated with menu actions.

3.8.1 Detailed Description

Manages the appearance and disappearance of a menu through animation and audio feedback.

3.8.2 Member Function Documentation

3.8.2.1 easeInSine()

```
float MenuHandler.easeInSine (
            float x )  [inline], [private]
```

Easing function for the animation, creating a smooth start effect.

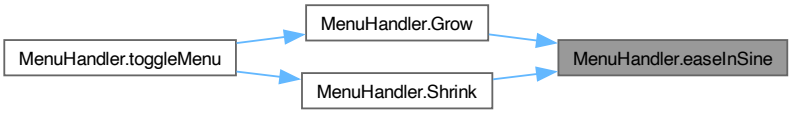
Parameters

x	The animation's progress as a fraction (0 to 1).

Returns

The eased progress value.

Here is the caller graph for this function:



3.8.2.2 Grow()

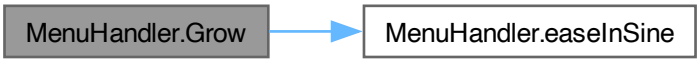
```
IEnumerator MenuHandler.Grow ( )  [inline], [private]
```

Coroutine to animate the menu growing to its target size.

Returns

IEnumerator for coroutine management.

Here is the call graph for this function:



Here is the caller graph for this function:

3.8.2.3 Shrink()

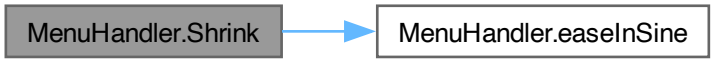
```
IEnumerator MenuHandler.Shrink ( )  [inline], [private]
```

Coroutine to animate the menu shrinking to disappear.

Returns

IEnumerator for coroutine management.

Here is the call graph for this function:



Here is the caller graph for this function:

3.8.2.4 Start()

```
void MenuHandler.Start ( )  [inline], [private]
```

Initializes the handler, finding the main camera.

3.8.2.5 toggleMenu()

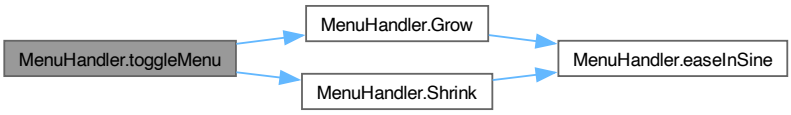
```
void MenuHandler.toggleMenu (
            bool forceClose )  [inline]
```

Toggles the menu's visibility, optionally forcing it to close, with animations and sound effects.

Parameters

forceClose	If true, forces the menu to close; otherwise, toggles the menu based on its current state.

Here is the call graph for this function:



3.8.3 Member Data Documentation

3.8.3.1 currentThread

```
Coroutine MenuHandler.currentThread  [private]
```

Coroutine reference for the currently running animation (growing or shrinking the menu).

3.8.3.2 dur

```
float MenuHandler.dur = 0.075f  [private]
```

Duration of the animation for opening or closing the menu.

3.8.3.3 handManager

```
HandManager MenuHandler.handManager  [private]
```

Reference to the HandManager to check and set the menu's enabled state.

3.8.3.4 Sounds

```
EventReference [] MenuHandler.Sounds  [private]
```

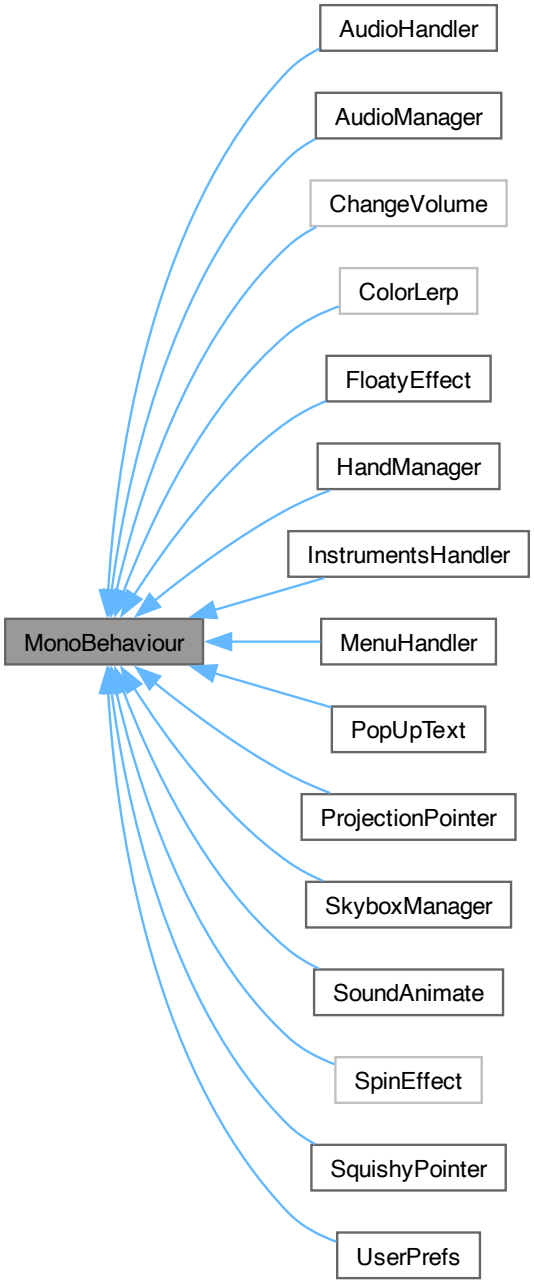
An array of EventReferences for playing sound effects associated with menu actions.

The documentation for this class was generated from the following file:

- Menu/MenuHandler.cs

3.9 MonoBehaviour Class Reference

Inheritance diagram for MonoBehaviour:



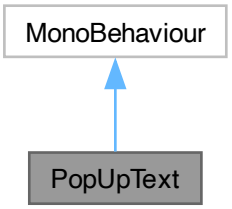
The documentation for this class was generated from the following file:

- Audio/AudioHandler.cs

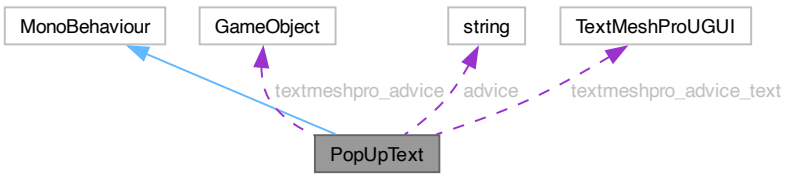
3.10 PopUpText Class Reference

Displays popup text on the screen by updating a TextMeshProUGUI component with provided advice text.

Inheritance diagram for PopUpText:



Collaboration diagram for PopUpText:



Public Attributes

- GameObject textmeshpro_advice

  The GameObject containing the TextMeshProUGUI component to be updated.
- string advice

  The string of advice to be displayed in the TextMeshProUGUI component.

Private Member Functions

- void Start ()

  Initializes the PopUpText by retrieving the TextMeshProUGUI component from the specified GameObject.
- void Update ()

  Updates the text displayed by the TextMeshProUGUI component each frame with the current advice string.

Private Attributes

- TextMeshProUGUI textmeshpro_advice_text

  The TextMeshProUGUI component where advice text will be displayed.

3.10.1 Detailed Description

Displays popup text on the screen by updating a TextMeshProUGUI component with provided advice text.

3.10.2 Member Function Documentation

3.10.2.1 Start()

```
void PopUpText.Start ( )  [inline], [private]
```

Initializes the PopUpText by retrieving the TextMeshProUGUI component from the specified GameObject.

3.10.2.2 Update()

```
void PopUpText.Update ( )  [inline], [private]
```

Updates the text displayed by the TextMeshProUGUI component each frame with the current advice string.

3.10.3 Member Data Documentation

3.10.3.1 advice

```
string PopUpText.advice
```

The string of advice to be displayed in the TextMeshProUGUI component.

3.10.3.2 textmeshpro_advice

```
GameObject PopUpText.textmeshpro_advice
```

The GameObject containing the TextMeshProUGUI component to be updated.

3.10.3.3 textmeshpro_advice_text

```
TextMeshProUGUI PopUpText.textmeshpro_advice_text  [private]
```

The TextMeshProUGUI component where advice text will be displayed.

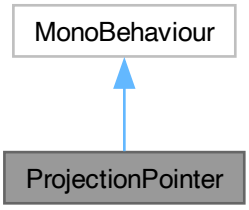
The documentation for this class was generated from the following file:

- Menu/PopUpText.cs

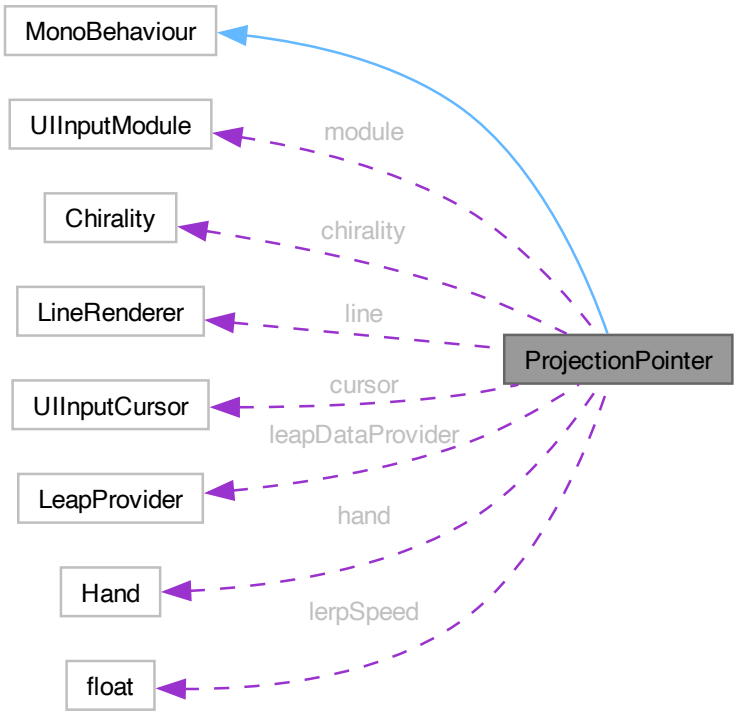
3.11 ProjectionPointer Class Reference

Manages the projection pointer for Leap Motion interaction, rendering a line between the hand and a UI cursor.

Inheritance diagram for ProjectionPointer:



Collaboration diagram for ProjectionPointer:



Public Attributes

- float **lerpSpeed** = 10

Private Member Functions

- void Awake ()

  Initializes the projection pointer by obtaining the Leap data provider from the UIInputModule. See UIInputModule for more information on this.
- void Update ()

  Updates the projection pointer, drawing a line from the hand to the UI cursor based on current Leap Motion data.

Private Attributes

- UIInputModule module

  Reference to the UIInputModule, used to obtain Leap data provider and interaction mode.
- Chirality chirality

  Specifies the hand chirality (left or right) to track with the projection pointer.
- LineRenderer line

  LineRenderer component used to draw the line between the hand and the cursor.
- UIInputCursor cursor

  Reference to the UIInputCursor, marking the target position for the projection pointer.
- LeapProvider **leapDataProvider**
- Hand hand

  Stores the current hand data from Leap Motion.

3.11.1 Detailed Description

Manages the projection pointer for Leap Motion interaction, rendering a line between the hand and a UI cursor.

3.11.2 Member Function Documentation

3.11.2.1 Awake()

```
void ProjectionPointer.Awake ( )  [inline], [private]
```

Initializes the projection pointer by obtaining the Leap data provider from the UIInputModule. See UIInputModule for more information on this.

3.11.2.2 Update()

```
void ProjectionPointer.Update ( )  [inline], [private]
```

Updates the projection pointer, drawing a line from the hand to the UI cursor based on current Leap Motion data.

3.11.3 Member Data Documentation

3.11.3.1 chirality

```
Chirality ProjectionPointer.chirality  [private]
```

Specifies the hand chirality (left or right) to track with the projection pointer.

3.11.3.2 cursor

```
UIInputCursor ProjectionPointer.cursor  [private]
```

Reference to the UIInputCursor, marking the target position for the projection pointer.

3.11.3.3 hand

```
Hand ProjectionPointer.hand  [private]
```

Stores the current hand data from Leap Motion.

3.11.3.4 line

```
LineRenderer ProjectionPointer.line  [private]
```

LineRenderer component used to draw the line between the hand and the cursor.

3.11.3.5 module

```
UIInputModule ProjectionPointer.module  [private]
```

Reference to the UIInputModule, used to obtain Leap data provider and interaction mode.

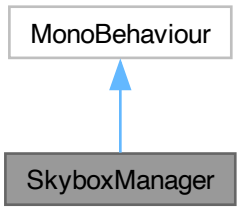
The documentation for this class was generated from the following file:

- Cursors/ProjectionPointer.cs

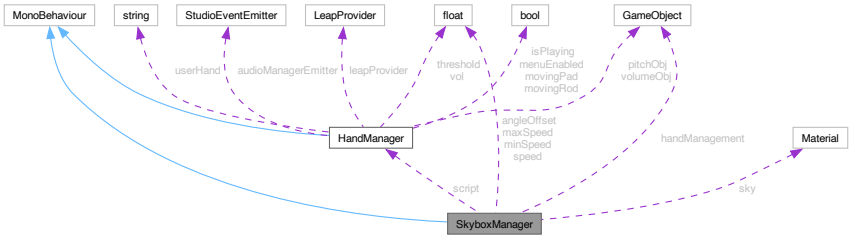
3.12 SkyboxManager Class Reference

Manages the rotation of the skybox in response to the music playing status from HandManager.

Inheritance diagram for SkyboxManager:



Collaboration diagram for SkyboxManager:



Private Member Functions

- void Awake ()

  Initializes the SkyboxManager by obtaining the HandManager component from the specified GameObject.
- void Update ()

  Updates the skybox's rotation speed and applies the rotation based on the music playing status.

Private Attributes

- HandManager script

  Reference to the HandManager script to check if music is playing.
- float **speed** = 0f
- float **angleOffset** = 10f
- float **maxSpeed** = 1f
- float **minSpeed** = 0.0f
- GameObject handManagement

  GameObject that manages the hand interactions.
- Material sky

  The skybox material to be rotated.

3.12.1 Detailed Description

Manages the rotation of the skybox in response to the music playing status from HandManager.

3.12.2 Member Function Documentation

3.12.2.1 Awake()

```
void SkyboxManager.Awake ( )  [inline], [private]
```

Initializes the SkyboxManager by obtaining the HandManager component from the specified GameObject.

3.12.2.2 Update()

```
void SkyboxManager.Update ( )  [inline], [private]
```

Updates the skybox's rotation speed and applies the rotation based on the music playing status.

3.12.3 Member Data Documentation

3.12.3.1 handManagement

```
GameObject SkyboxManager.handManagement  [private]
```

GameObject that manages the hand interactions.

3.12.3.2 script

```
HandManager SkyboxManager.script  [private]
```

Reference to the HandManager script to check if music is playing.

3.12.3.3 sky

```
Material SkyboxManager.sky  [private]
```

The skybox material to be rotated.

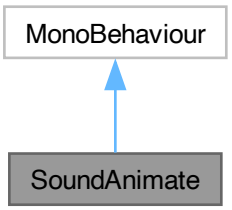
The documentation for this class was generated from the following file:

- Other/SkyboxManager.cs

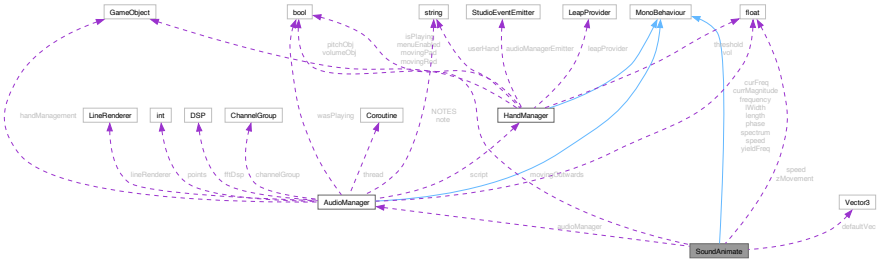
3.13 SoundAnimate Class Reference

Animates a GameObject's position in response to audio amplitude, creating a visual representation of sound intensity.

Inheritance diagram for SoundAnimate:



Collaboration diagram for SoundAnimate:



Public Attributes

- AudioManager audioManager

  Reference to the AudioManager, used to obtain the current amplitude of playing audio.
- float **speed** = 0.0f

Private Member Functions

- void Start ()

  Initializes the SoundAnimate script by recording the GameObject's original local position.
- void Update ()

  Updates the GameObject's position based on audio amplitude, creating a forward and backward animation effect.

Private Attributes

- Vector3 defaultVec

  The GameObject's original local position, used as a base for animation.
- float **zMovement**
- bool movingOutwards = true

  Flag indicating whether the GameObject is currently moving outwards or returning to its original position.

3.13.1 Detailed Description

Animates a GameObject's position in response to audio amplitude, creating a visual representation of sound intensity.

3.13.2 Member Function Documentation

3.13.2.1 Start()

```
void SoundAnimate.Start ( )  [inline], [private]
```

Initializes the SoundAnimate script by recording the GameObject's original local position.

3.13.2.2 Update()

```
void SoundAnimate.Update ( )  [inline], [private]
```

Updates the GameObject's position based on audio amplitude, creating a forward and backward animation effect.

3.13.3 Member Data Documentation

3.13.3.1 audioManager

```
AudioManager SoundAnimate.audioManager
```

Reference to the AudioManager, used to obtain the current amplitude of playing audio.

3.13.3.2 defaultVec

```
Vector3 SoundAnimate.defaultVec  [private]
```

The GameObject's original local position, used as a base for animation.

3.13.3.3 movingOutwards

```
bool SoundAnimate.movingOutwards = true  [private]
```

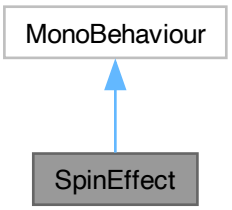
Flag indicating whether the GameObject is currently moving outwards or returning to its original position.

The documentation for this class was generated from the following file:

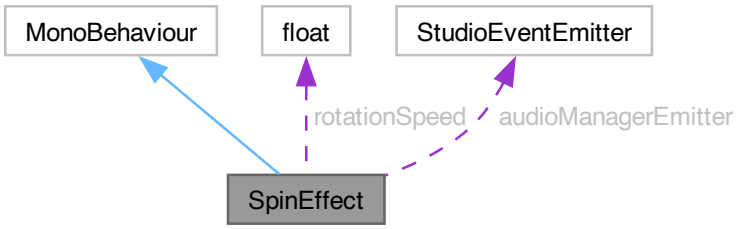
- Other/SoundAnimate.cs

3.14 SpinEffect Class Reference

Inheritance diagram for SpinEffect:



Collaboration diagram for SpinEffect:



Public Attributes

- float **rotationSpeed** = 50f

Private Member Functions

- void **Start** ()
- void **Update** ()

Private Attributes

- StudioEventEmitter **audioManagerEmitter**

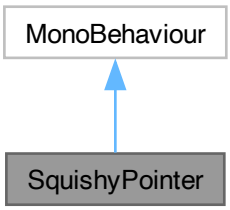
The documentation for this class was generated from the following file:

- Other/SpinEffect.cs

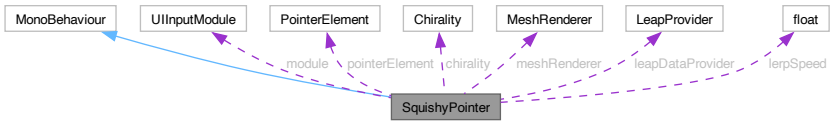
3.15 SquishyPointer Class Reference

Represents a squishy pointer that visualizes hand pinching actions using Leap Motion.

Inheritance diagram for SquishyPointer:



Collaboration diagram for SquishyPointer:



Public Attributes

- float **lerpSpeed** = 10

Private Member Functions

- void Awake ()

  Initializes the squishy pointer by setting up Leap Motion data provider and initializing the pointer material.
- void Update ()

Private Attributes

- UIInputModule module

  Reference to the UIInputModule, used to obtain the Leap data provider.
- PointerElement pointerElement

  PointerElement that the squishy pointer will point towards.
- Chirality chirality

  Specifies the hand chirality (left or right) to track with the squishy pointer.
- MeshRenderer **meshRenderer**
- LeapProvider **leapDataProvider**

3.15.1 Detailed Description

Represents a squishy pointer that visualizes hand pinching actions using Leap Motion.

This class is responsible for adjusting the position, scale, and color of a pointer based on the user's hand pinch strength.

3.15.2 Member Function Documentation

3.15.2.1 Awake()

```
void SquishyPointer.Awake ( )  [inline], [private]
```

Initializes the squishy pointer by setting up Leap Motion data provider and initializing the pointer material.

3.15.2.2 Update()

```
void SquishyPointer.Update ( )  [inline], [private]
```

The pointer interpolates towards the hand's pinch position, changes scale based on pinch strength, and updates its color from white to green based on the pinch strength.

3.15.3 Member Data Documentation

3.15.3.1 chirality

```
Chirality SquishyPointer.chirality  [private]
```

Specifies the hand chirality (left or right) to track with the squishy pointer.

3.15.3.2 module

```
UIInputModule SquishyPointer.module  [private]
```

Reference to the UIInputModule, used to obtain the Leap data provider.

3.15.3.3 pointerElement

```
PointerElement SquishyPointer.pointerElement  [private]
```

PointerElement that the squishy pointer will point towards.

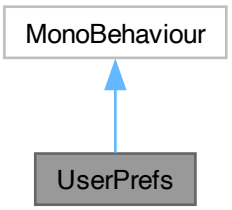
The documentation for this class was generated from the following file:

- Cursors/SquishyPointer.cs

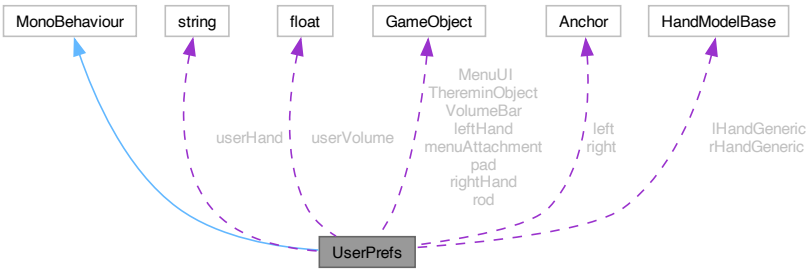
3.16 UserPrefs Class Reference

Manages user preferences for a VR application, including hand dominance and volume settings.

Inheritance diagram for UserPrefs:



Collaboration diagram for UserPrefs:



Public Member Functions

- delegate void PreferenceChangedEventHandler (string key, object value)

  Delegate for handling preference change events.
- void updateValues ()

  Updates the user volume preference based on interaction with the volume bar.
- void setHand (string Hand)

  Sets the user's hand dominance, updating UI elements and interaction anchors accordingly.

Events

- static PreferenceChangedEventHandler OnPreferenceChanged

  Event triggered when a preference changes.

Private Member Functions

- void Start ()

  Initializes user preferences, setting default values if none are found.
- void swapAnchors (string Hand)

  Swaps the positions of the rod and pad based on the user's hand dominance.

Private Attributes

- string userHand

  Stores the user's hand preference.
- float userVolume

  Stores the user's preferred volume level.
- GameObject **rod**
- GameObject **pad**
- Anchor left

  Anchors for left and right hand positions.
- Anchor **right**
- GameObject **MenuUI**
- GameObject **VolumeBar**
- GameObject **leftHand**
- GameObject **rightHand**
- HandModelBase **lHandGeneric**
- HandModelBase **rHandGeneric**
- GameObject **menuAttachment**
- GameObject **ThereminObject**

3.16.1 Detailed Description

Manages user preferences for a VR application, including hand dominance and volume settings.

3.16.2 Member Function Documentation

3.16.2.1 PreferenceChangedEventHandler()

```
delegate void UserPrefs.PreferenceChangedEventHandler (
            string key,
            object value )
```

Delegate for handling preference change events.

3.16.2.2 setHand()

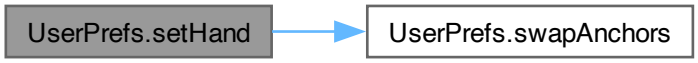
```
void UserPrefs.setHand (
            string Hand )  [inline]
```

Sets the user's hand dominance, updating UI elements and interaction anchors accordingly.

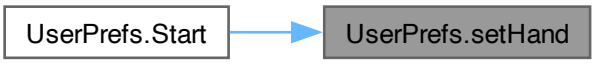
Parameters

Hand	The hand dominance (either "Left" or "Right").

Here is the call graph for this function:



Here is the caller graph for this function:



3.16.2.3 Start()

```
void UserPrefs.Start ( )  [inline], [private]
```

Initializes user preferences, setting default values if none are found.

This sets by default the following settings to their resepects values UserHand: Right UserVolume: 0.5F Master↩ Volume: userVolume Here is the call graph for this function:

3.16.2.4 swapAnchors()

```
void UserPrefs.swapAnchors (
            string Hand )  [inline], [private]
```

Swaps the positions of the rod and pad based on the user's hand dominance.

Parameters

Hand	The hand dominance to swap anchors for.

Here is the caller graph for this function:

3.16.2.5 updateValues()

```
void UserPrefs.updateValues ( )  [inline]
```

Updates the user volume preference based on interaction with the volume bar.

3.16.3 Member Data Documentation

3.16.3.1 left

```
Anchor UserPrefs.left  [private]
```

Anchors for left and right hand positions.

3.16.3.2 userHand

```
string UserPrefs.userHand  [private]
```

Stores the user's hand preference.

3.16.3.3 userVolume

```
float UserPrefs.userVolume  [private]
```

Stores the user's preferred volume level.

3.16.4 Event Documentation

3.16.4.1 OnPreferenceChanged

```
PreferenceChangedEventHandler UserPrefs.OnPreferenceChanged  [static]
```

Event triggered when a preference changes.

The documentation for this class was generated from the following file:

- UserPrefs.cs

Index