OSSC

0.0.4

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Documentation

1.1 OSSC Namespace Reference

Namespaces

- namespace Editor
- namespace Model

Classes

- class CueManager

  Manages all SoundCues
- interface ISoundCue

  SoundCue Inteface. SoundController returns a SoundCue Interface to further control the playing SouncCue
- struct PlaySoundSettings

  Set the settings to play a particular cue with particular preferences.
- class SoundController

  The main class that is used for Playing and controlling all sounds.
- class SoundCue

  Plays a whole cue of soundItems
- struct SoundCueData

  Used for sending data to play to AudioCue
- class SoundCueProxy

  Given by the SoundController to User as ISoundCue to control the playing SoundCue.
- class SoundObject

  Used by the SoundCue. Controls the AudioSource.
- class SoundTags

  Used By the SoundController for tagging SoundItems.
- struct TagData

  Used by the SoundTags to save Tags.

1.2 OSSC.Editor Namespace Reference

Classes

- class SoundControllerEditor

  Draws the Custom Editor for SoundController
- class SoundObjectEditor

  Draw the custom editor inspector for SoundObject

1.3 OSSC.Model Namespace Reference

Classes

- class CategoryItem

  Used by the SoundControllerData to store categories.
- class CustomRange

  Used by SoundItem to store Random Ranges.
- class SoundControllerData

  SoundController's Database.
- class SoundItem

  Used by CategoryItem to store sounds data.

Chapter 2

Class Documentation

2.1 OSSC.Model.CategoryItem Class Reference

Used by the SoundControllerData to store categories.

Public Attributes

- string name

    Category name
- SoundItem [ ] soundItems

    Array of SoundItems
- GameObject audioObjectPrefab

    Alternative SoundObject prefab to use, instead of the Default one from SoundController.
- bool usingDefaultPrefab = true

    Check whether to use alternative SoundObject prefab.
- float categoryVolume = 1f

    Volume of the category
- bool foldOutSoundItems = false

    Used for Editor to save whether the SoundItems are folded out or not.
- string soundsSearchName = ""

    Save the last search name written in editor.
- bool isMute = false

    Is Category mute?

2.1.1 Detailed Description

Used by the SoundControllerData to store categories.

Definition at line 11 of file CategoryItem.cs.

2.1.2 Member Data Documentation

2.1.2.1 audioObjectPrefab

```
GameObject OSSC.Model.CategoryItem.audioObjectPrefab
```

Alternative SoundObject prefab to use, instead of the Default one from SoundController.

Definition at line 24 of file CategoryItem.cs.

2.1.2.2 categoryVolume

```
float OSSC.Model.CategoryItem.categoryVolume = 1f
```

Volume of the category

Definition at line 34 of file CategoryItem.cs.

2.1.2.3 foldOutSoundItems

```
bool OSSC.Model.CategoryItem.foldOutSoundItems = false
```

Used for Editor to save whether the SoundItems are folded out or not.

Definition at line 39 of file CategoryItem.cs.

2.1.2.4 isMute

```
bool OSSC.Model.CategoryItem.isMute = false
```

Is Category mute?

Definition at line 47 of file CategoryItem.cs.

2.1.2.5 name

```
string OSSC.Model.CategoryItem.name
```

Category name

Definition at line 16 of file CategoryItem.cs.

2.1.2.6 soundItems

```
SoundItem [] OSSC.Model.CategoryItem.soundItems
```

Array of SoundItems

Definition at line 20 of file CategoryItem.cs.

2.1.2.7 soundsSearchName

```
string OSSC.Model.CategoryItem.soundsSearchName = ""
```

Save the last search name written in editor.

Definition at line 43 of file CategoryItem.cs.

2.1.2.8 usingDefaultPrefab

```
bool OSSC.Model.CategoryItem.usingDefaultPrefab = true
```

Check whether to use alternative SoundObject prefab.

Definition at line 28 of file CategoryItem.cs.

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

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Model/CategoryItem.cs

2.2 OSSC.CueManager Class Reference

Manages all SoundCues

Public Member Functions

- CueManager ()
  Default Constructor
- CueManager (int initialSize)
  Costruct CueManager with some initial SoundCues created.
- SoundCue GetSoundCue ()
  Get a free SoundCue.
- void StopAllCues (bool shouldCallOnEndCallback=true)
  Stops all SoundCues from playing.

2.2.1 Detailed Description

Manages all SoundCues

Definition at line 10 of file CueManager.cs.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 CueManager() [1/2]

```
OSSC.CueManager.CueManager ( )
```

Default Constructor

Definition at line 23 of file CueManager.cs.

2.2.2.2 CueManager() [2/2]

```
OSSC.CueManager.CueManager (
             int initialSize )
```

Costruct CueManager with some initial SoundCues created.

Parameters

initialSize	Size of the SoundCue pool.

Definition at line 32 of file CueManager.cs.

2.2.3 Member Function Documentation

2.2.3.1 GetSoundCue()

```
SoundCue OSSC.CueManager.GetSoundCue ( )
```

Get a free SoundCue.

Returns

Returns a SoundCue instance.

Definition at line 41 of file CueManager.cs.

2.2.3.2 StopAllCues()

```
void OSSC.CueManager.StopAllCues (
            bool shouldCallOnEndCallback = true )
```

Stops all SoundCues from playing.

Parameters

shouldCallOnEndCallback	Check whether to call OnEnd events or not.

Definition at line 52 of file CueManager.cs.

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

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/CueManager.cs

2.3 OSSC.Model.CustomRange Class Reference

Used by SoundItem to store Random Ranges.

Public Member Functions

- float GetRandomRange ()

  Gets a random value from it's Minimum and Maximum limits.

Public Attributes

- float min = 1f

  Minimum limit
- float max = 1f

  Maximum limit

2.3.1 Detailed Description

Used by SoundItem to store Random Ranges.

Definition at line 55 of file SoundItem.cs.

2.3.2 Member Function Documentation

2.3.2.1 GetRandomRange()

```
float OSSC.Model.CustomRange.GetRandomRange ( )
```

Gets a random value from it's Minimum and Maximum limits.

Returns

Definition at line 70 of file SoundItem.cs.

2.3.3 Member Data Documentation

2.3.3.1 max

```
float OSSC.Model.CustomRange.max = 1f
```

Maximum limit

Definition at line 64 of file SoundItem.cs.

2.3.3.2 min

```
float OSSC.Model.CustomRange.min = 1f
```

Minimum limit

Definition at line 60 of file SoundItem.cs.

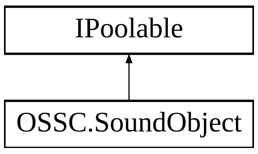
The documentation for this class was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Model/SoundItem.cs

2.4 IPoolable Interface Reference

Used by the ObjectPool

Inheritance diagram for IPoolable:



Public Member Functions

- bool IsFree ()

  Checks whether the poolable object is free.

Properties

- PrefabBasedPool pool `[get, set]`

  Saves the pool that it belongs to.

2.4.1 Detailed Description

Used by the ObjectPool

Definition at line 8 of file IPoolable.cs.

2.4.2 Member Function Documentation

2.4.2.1 IsFree()

```
bool IPoolable.IsFree ( )
```

Checks whether the poolable object is free.

Returns

True - is Free, False - is busy

Implemented in OSSC.SoundObject.

2.4.3 Property Documentation

2.4.3.1 pool

```
PrefabBasedPool IPoolable.pool  [get], [set]
```

Saves the pool that it belongs to.

Definition at line 13 of file IPoolable.cs.

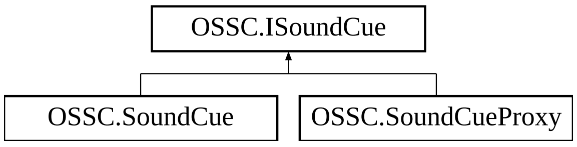
The documentation for this interface was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/IPoolable.cs

2.5 OSSC.ISoundCue Interface Reference

SoundCue Inteface. SoundController returns a SoundCue Interface to further control the playing SouncCue

Inheritance diagram for OSSC.ISoundCue:



Public Member Functions

- void Play (SoundCueData data)

  Plays the SoundCue. This method is called by the SoundController.
- void Pause ()

  Pause the SoundCue.
- void Resume ()

  Resume the paused SoundCue.
- void Stop (bool shouldCallOnFinishedCue=true)

  Stop the SoundCue from playing.

Properties

- Action$<$ string $>$ OnPlayEnded `[get, set]`

  Called everytime a SoundItem finished playing in SoundCue.
- Action$<$ SoundCue $>$ OnPlayCueEnded `[get, set]`

  Called everytime a SoundCue finished playing.
- SoundObject AudioObject `[get, set]`

  Used by the SoundCue to play all SoundItems.
- SoundCueData Data `[get]`

  Data collected by the Soundcontroller. Has all SoundItems that needs to be played.
- bool IsPlaying `[get]`

  Check if SoundCue is still playing.
- int ID `[get]`

  SoundCue Identifier

2.5.1 Detailed Description

SoundCue Inteface. SoundController returns a SoundCue Interface to further control the playing SouncCue

Definition at line 12 of file ISoundCue.cs.

2.5.2 Member Function Documentation

2.5.2.1 Pause()

```
void OSSC.ISoundCue.Pause ( )
```

Pause the SoundCue.

Implemented in OSSC.SoundCueProxy, and OSSC.SoundCue.

2.5.2.2 Play()

```
void OSSC.ISoundCue.Play (
            SoundCueData data )
```

Plays the SoundCue. This method is called by the SoundController.

Parameters

data	The Data needed for the SoundCue to play.

Implemented in OSSC.SoundCueProxy, and OSSC.SoundCue.

2.5.2.3 Resume()

```
void OSSC.ISoundCue.Resume ( )
```

Resume the paused SoundCue.

Implemented in OSSC.SoundCueProxy, and OSSC.SoundCue.

2.5.2.4 Stop()

```
void OSSC.ISoundCue.Stop (
            bool shouldCallOnFinishedCue = true )
```

Stop the SoundCue from playing.

Parameters

shouldCallOnFinishedCue	Select whether to Call OnEnd events or not.

Implemented in OSSC.SoundCueProxy, and OSSC.SoundCue.

2.5.3 Property Documentation

2.5.3.1 AudioObject

`SoundObject OSSC.ISoundCue.AudioObject [get], [set]`

Used by the SoundCue to play all SoundItems.

Definition at line 25 of file ISoundCue.cs.

2.5.3.2 Data

`SoundCueData OSSC.ISoundCue.Data [get]`

Data collected by the Soundcontroller. Has all SoundItems that needs to be played.

Definition at line 29 of file ISoundCue.cs.

2.5.3.3 ID

`int OSSC.ISoundCue.ID [get]`

SoundCue Identifier

Definition at line 37 of file ISoundCue.cs.

2.5.3.4 IsPlaying

`bool OSSC.ISoundCue.IsPlaying [get]`

Check if SoundCue is still playing.

Definition at line 33 of file ISoundCue.cs.

2.5.3.5 OnPlayCueEnded

`Action<SoundCue> OSSC.ISoundCue.OnPlayCueEnded [get], [set]`

Called everytime a SoundCue finished playing.

Definition at line 21 of file ISoundCue.cs.

2.5.3.6 OnPlayEnded

```
Action<string> OSSC.ISoundCue.OnPlayEnded [get], [set]
```

Called everytime a SoundItem finished playing in SoundCue.

Definition at line 17 of file ISoundCue.cs.

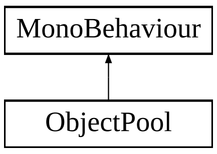
The documentation for this interface was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/ISoundCue.cs

2.6 ObjectPool Class Reference

Creates a pool of different prefabs when someone requests a GameObject.

Inheritance diagram for ObjectPool:



Public Member Functions

- GameObject GetFreeObject (GameObject prefab=null)
  Gets a Free GameObject.

Public Attributes

- List< PrefabBasedPool > pools
  The list of Prefab based pools

2.6.1 Detailed Description

Creates a pool of different prefabs when someone requests a GameObject.

Definition at line 8 of file ObjectPool.cs.

2.6.2 Member Function Documentation

2.6.2.1 GetFreeObject()

```
GameObject ObjectPool.GetFreeObject (
            GameObject prefab = null )
```

Gets a Free GameObject.

Parameters

prefab	The kind of GameObject to return

Returns

Returns the requested GameObject instance

Definition at line 23 of file ObjectPool.cs.

2.6.3 Member Data Documentation

2.6.3.1 pools

```
List<PrefabBasedPool> ObjectPool.pools
```

The list of Prefab based pools

Definition at line 14 of file ObjectPool.cs.

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

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/ObjectPool.cs

2.7 OSSC.PlaySoundSettings Struct Reference

Set the settings to play a particular cue with particular preferences.

Public Member Functions

- void Init ()

  Initializes the PlaySoundSettings with predefined values. It is required to be called after the creation of the Play↩ SoundSettings instance.

Public Attributes

- string name

  Name of the soundItem to be played
- string [ ] names

  A list of sound Items to be played consecutively
- Transform parent

  Attach the Playing sound to a Specific GameObject
- float fadeInTime

  Fade In time of the whole SoundCue
- float fadeOutTime

  Fade Out time of the whole SoundCue
- string categoryName

  Play SoundItems from a specific Category
- bool isLooped

  Control whether the SoundCue should loop
- ISoundCue soundCueProxy

  Use the same SoundCue to play again the sounds played in that SoundCue This is recommended to do, because searching by names all the Sounds to play is very expensive.
- string tagName

  Play soundItems that correspond to the tag

2.7.1 Detailed Description

Set the settings to play a particular cue with particular preferences.

Definition at line 324 of file SoundController.cs.

2.7.2 Member Function Documentation

2.7.2.1 Init()

```
void OSSC.PlaySoundSettings.Init ( )
```

Initializes the PlaySoundSettings with predefined values. It is required to be called after the creation of the Play↩ SoundSettings instance.

Definition at line 368 of file SoundController.cs.

2.7.3 Member Data Documentation

2.7.3.1 categoryName

```
string OSSC.PlaySoundSettings.categoryName
```

Play SoundItems from a specific Category

Definition at line 349 of file SoundController.cs.

2.7.3.2 fadeInTime

```
float OSSC.PlaySoundSettings.fadeInTime
```

Fade In time of the whole SoundCue

Definition at line 341 of file SoundController.cs.

2.7.3.3 fadeOutTime

```
float OSSC.PlaySoundSettings.fadeOutTime
```

Fade Out time of the whole SoundCue

Definition at line 345 of file SoundController.cs.

2.7.3.4 isLooped

```
bool OSSC.PlaySoundSettings.isLooped
```

Control whether the SoundCue should loop

Definition at line 353 of file SoundController.cs.

2.7.3.5 name

```
string OSSC.PlaySoundSettings.name
```

Name of the soundItem to be played

Definition at line 329 of file SoundController.cs.

2.7.3.6 names

```
string [] OSSC.PlaySoundSettings.names
```

A list of sound Items to be played consecutively

Definition at line 333 of file SoundController.cs.

2.7.3.7 parent

```
Transform OSSC.PlaySoundSettings.parent
```

Attach the Playing sound to a Specific GameObject

Definition at line 337 of file SoundController.cs.

2.7.3.8 soundCueProxy

```
ISoundCue OSSC.PlaySoundSettings.soundCueProxy
```

Use the same SoundCue to play again the sounds played in that SoundCue This is recommended to do, because searching by names all the Sounds to play is very expensive.

Definition at line 358 of file SoundController.cs.

2.7.3.9 tagName

```
string OSSC.PlaySoundSettings.tagName
```

Play soundItems that correspond to the tag

Definition at line 362 of file SoundController.cs.

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

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundController.cs

2.8 PrefabBasedPool Class Reference

Public Member Functions

- PrefabBasedPool (GameObject prefab)
- GameObject GetFreeObject ()
- void Despawn (GameObject obj)

Public Attributes

- GameObject prefab
- List< GameObject > pool
- Transform parent

  Where pooled objects will reside.

2.8.1 Detailed Description

Definition at line 58 of file ObjectPool.cs.

2.8.2 Constructor & Destructor Documentation

2.8.2.1 PrefabBasedPool()

```
PrefabBasedPool.PrefabBasedPool (
              GameObject prefab )
```

Definition at line 60 of file ObjectPool.cs.

2.8.3 Member Function Documentation

2.8.3.1 Despawn()

```
void PrefabBasedPool.Despawn (
              GameObject obj )
```

Definition at line 100 of file ObjectPool.cs.

2.8.3.2 GetFreeObject()

```
GameObject PrefabBasedPool.GetFreeObject ( )
```

Definition at line 73 of file ObjectPool.cs.

2.8.4 Member Data Documentation

2.8.4.1 parent

```
Transform PrefabBasedPool.parent
```

Where pooled objects will reside.

Definition at line 71 of file ObjectPool.cs.

2.8.4.2 pool

```
List<GameObject> PrefabBasedPool.pool
```

Definition at line 66 of file ObjectPool.cs.

2.8.4.3 prefab

```
GameObject PrefabBasedPool.prefab
```

Definition at line 65 of file ObjectPool.cs.

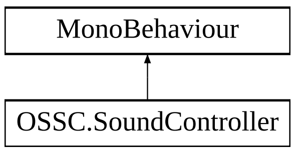
The documentation for this class was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/ObjectPool.cs

2.9 OSSC.SoundController Class Reference

The main class that is used for Playing and controlling all sounds.

Inheritance diagram for OSSC.SoundController:



Public Member Functions

- void StopAll (bool shouldCallOnEndCallback=true)

  Stop all Playing Sound Cues.
- void SetMute (string categoryName, bool value)

  Set mute a category.
- ISoundCue Play (PlaySoundSettings settings)

  Creates a SoundCue and plays it.

Public Attributes

- GameObject _defaultPrefab

  Default prefab with SoundObject and AudioSource. It is used by the Soundcontroller to play SoundCues.
- SoundControllerData _database

  Saves all the data that the SoundController uses.

Properties

- GameObject defaultPrefab `[set]`

  Set the default Prefab with SoundObject and AudioSource in it.

2.9.1 Detailed Description

The main class that is used for Playing and controlling all sounds.

Definition at line 12 of file SoundController.cs.

2.9.2 Member Function Documentation

2.9.2.1 Play()

```
ISoundCue OSSC.SoundController.Play (
            PlaySoundSettings settings )
```

Creates a SoundCue and plays it.

Parameters

settings	A struct which contains all data for SoundController to work

Returns

A soundCue interface which can be subscribed to it's events.

Definition at line 87 of file SoundController.cs.

2.9.2.2 SetMute()

```
void OSSC.SoundController.SetMute (
            string categoryName,
            bool value )
```

Set mute a category.

Parameters

categoryName	Name of the cateogory
value	True to mute, false to unmute

Definition at line 71 of file SoundController.cs.

2.9.2.3 StopAll()

```
void OSSC.SoundController.StopAll (
             bool shouldCallOnEndCallback = true )
```

Stop all Playing Sound Cues.

Parameters

shouldCallOnEndCallback	Control whether to call the OnEnd event, or not.

Definition at line 61 of file SoundController.cs.

2.9.3 Member Data Documentation

2.9.3.1 _database

```
SoundControllerData OSSC.SoundController._database
```

Saves all the data that the SoundController uses.

Definition at line 23 of file SoundController.cs.

2.9.3.2 _defaultPrefab

```
GameObject OSSC.SoundController._defaultPrefab
```

Default prefab with SoundObject and AudioSource. It is used by the Soundcontroller to play SoundCues.

Definition at line 19 of file SoundController.cs.

2.9.4 Property Documentation

2.9.4.1 defaultPrefab

```
GameObject OSSC.SoundController.defaultPrefab  [set]
```

Set the default Prefab with SoundObject and AudioSource in it.

Definition at line 50 of file SoundController.cs.

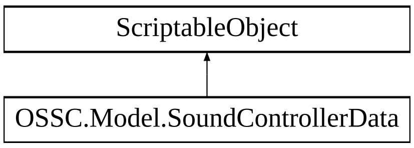
The documentation for this class was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundController.cs

2.10 OSSC.Model.SoundControllerData Class Reference

SoundController's Database.

Inheritance diagram for OSSC.Model.SoundControllerData:



Public Attributes

- CategoryItem [ ] items

  Stores all created Categories.
- bool foldOutCategories = false

  Checks in editor whether the categories should fold out or not.
- bool foldOutTags = false

  check if editor should fold out the tags or not.
- string assetName

  Database name.
- SoundTags soundTags

  Stores the Created tags from Editor.

2.10.1 Detailed Description

SoundController's Database.

Definition at line 11 of file SoundControllerData.cs.

2.10.2 Member Data Documentation

2.10.2.1 assetName

```
string OSSC.Model.SoundControllerData.assetName
```

Database name.

Definition at line 28 of file SoundControllerData.cs.

2.10.2.2 foldOutCategories

```
bool OSSC.Model.SoundControllerData.foldOutCategories = false
```

Checks in editor whether the categories should fold out or not.

Definition at line 20 of file SoundControllerData.cs.

2.10.2.3 foldOutTags

```
bool OSSC.Model.SoundControllerData.foldOutTags = false
```

check if editor should fold out the tags or not.

Definition at line 24 of file SoundControllerData.cs.

2.10.2.4 items

```
CategoryItem [] OSSC.Model.SoundControllerData.items
```

Stores all created Categories.

Definition at line 16 of file SoundControllerData.cs.

2.10.2.5 soundTags

```
SoundTags OSSC.Model.SoundControllerData.soundTags
```

Stores the Created tags from Editor.

Definition at line 32 of file SoundControllerData.cs.

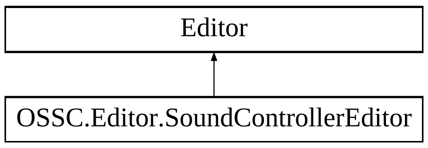
The documentation for this class was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Model/SoundControllerData.cs

2.11 OSSC.Editor.SoundControllerEditor Class Reference

Draws the Custom Editor for SoundController

Inheritance diagram for OSSC.Editor.SoundControllerEditor:



Public Member Functions

- override void OnInspectorGUI ()

    Draws the Inspector GUI

2.11.1 Detailed Description

Draws the Custom Editor for SoundController

Definition at line 16 of file SoundControllerEditor.cs.

2.11.2 Member Function Documentation

2.11.2.1 OnInspectorGUI()

```
override void OSSC.Editor.SoundControllerEditor.OnInspectorGUI ( )
```

Draws the Inspector GUI

Definition at line 46 of file SoundControllerEditor.cs.

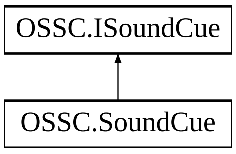
The documentation for this class was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Editor/SoundControllerEditor.cs

2.12 OSSC.SoundCue Class Reference

Plays a whole cue of soundItems

Inheritance diagram for OSSC.SoundCue:



Public Member Functions

- SoundCue ()

  Default Constructor
- SoundCue (int id)

  Custom Constructor
- void Play (SoundCueData data)

  Will start playing the cue. NOTE: It is called from SoundCueProxy that is created by the SoundController.
- void Play (SoundCueData data, SoundCueProxy proxy)

  Plays the SoundCue.
- void Pause ()

  Will pause the cue;
- void Resume ()

  Resume the cue from where it was paused.
- void Stop (bool shouldCallOnFinishedCue=true)

  Stops the SoundCue.

Properties

- Action< string > OnPlayEnded `[get, set]`

  Check ISoundCue
- Action< SoundCue > OnPlayCueEnded `[get, set]`

  Check ISoundCue
- Action< SoundCue, SoundCueProxy > OnPlayKilled `[get, set]`

  Called whenever the sound cue has finished playing or was stopped
- SoundObject AudioObject `[get, set]`

  Check ISoundCue
- SoundCueData Data `[get]`

  Check ISoundCue
- bool IsPlaying `[get]`

  Check ISoundCue
- int ID `[get]`

  SoundCue's unique ID given by the manager

2.12.1 Detailed Description

Plays a whole cue of soundItems

Definition at line 12 of file SoundCue.cs.

2.12.2 Constructor & Destructor Documentation

2.12.2.1 SoundCue() [1/2]

```
OSSC.SoundCue.SoundCue ( )
```

Default Constructor

Definition at line 61 of file SoundCue.cs.

2.12.2.2 SoundCue() [2/2]

```
OSSC.SoundCue.SoundCue (
             int id )
```

Custom Constructor

Parameters

id	Sets the ID of the SoundCue.

Definition at line 69 of file SoundCue.cs.

2.12.3 Member Function Documentation

2.12.3.1 Pause()

```
void OSSC.SoundCue.Pause ( )
```

Will pause the cue;

Implements OSSC.ISoundCue.

Definition at line 118 of file SoundCue.cs.

2.12.3.2 Play() [1/2]

```
void OSSC.SoundCue.Play (
             SoundCueData data )
```

Will start playing the cue. NOTE: It is called from SoundCueProxy that is created by the SoundController.

Implements OSSC.ISoundCue.

Definition at line 91 of file SoundCue.cs.

2.12.3.3 Play() [2/2]

```
void OSSC.SoundCue.Play (
            SoundCueData data,
            SoundCueProxy proxy )
```

Plays the SoundCue.

Parameters

data	SoundCue's data
proxy	Proxy created by SoundController that called this method.

Definition at line 109 of file SoundCue.cs.

2.12.3.4 Resume()

```
void OSSC.SoundCue.Resume ( )
```

Resume the cue from where it was paused.

Implements OSSC.ISoundCue.

Definition at line 127 of file SoundCue.cs.

2.12.3.5 Stop()

```
void OSSC.SoundCue.Stop (
             bool shouldCallOnFinishedCue = true )
```

Stops the SoundCue.

Parameters

shouldCallOnFinishedCue	Checks whether to call OnEnd events, or not.

Implements OSSC.ISoundCue.

Definition at line 137 of file SoundCue.cs.

2.12.4 Property Documentation

2.12.4.1 AudioObject

```
SoundObject OSSC.SoundCue.AudioObject  [get], [set]
```

Check ISoundCue

Definition at line 32 of file SoundCue.cs.

2.12.4.2 Data

```
SoundCueData OSSC.SoundCue.Data  [get]
```

Check ISoundCue

Definition at line 37 of file SoundCue.cs.

2.12.4.3 ID

```
int OSSC.SoundCue.ID  [get]
```

SoundCue's unique ID given by the manager

Returns

Definition at line 53 of file SoundCue.cs.

2.12.4.4 IsPlaying

```
bool OSSC.SoundCue.IsPlaying  [get]
```

Check ISoundCue

Definition at line 43 of file SoundCue.cs.

2.12.4.5 OnPlayCueEnded

```
Action<SoundCue> OSSC.SoundCue.OnPlayCueEnded  [get], [set]
```

Check ISoundCue

Definition at line 22 of file SoundCue.cs.

2.12.4.6 OnPlayEnded

```
Action<string> OSSC.SoundCue.OnPlayEnded  [get], [set]
```

Check ISoundCue

Definition at line 17 of file SoundCue.cs.

2.12.4.7 OnPlayKilled

```
Action<SoundCue, SoundCueProxy> OSSC.SoundCue.OnPlayKilled  [get], [set]
```

Called whenever the sound cue has finished playing or was stopped

Definition at line 27 of file SoundCue.cs.

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

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundCue.cs

2.13 OSSC.SoundCueData Struct Reference

Used for sending data to play to AudioCue

Public Attributes

- SoundItem [ ] sounds

    sound items that played by the SoundCue.
- CategoryItem [ ] categoriesForSounds

    category items that correspond with each of SoundItem in sounds.
- float [ ] categoryVolumes

    Category sound volumes that correspond with Sound items.
- GameObject audioPrefab

    Prefab with SoundObject to play Sound items.
- float fadeInTime

    Fade In time.
- float fadeOutTime

    Fade Out time.
- bool isFadeIn

    Should SoundCue Fade In?
- bool isFadeOut

    Should SoundCue Fade Out?
- bool isLooped

    Should SoundCue be looped?

2.13.1 Detailed Description

Used for sending data to play to AudioCue

Definition at line 282 of file SoundCue.cs.

2.13.2 Member Data Documentation

2.13.2.1 audioPrefab

```
GameObject OSSC.SoundCueData.audioPrefab
```

Prefab with SoundObject to play Sound items.

Definition at line 299 of file SoundCue.cs.

2.13.2.2 categoriesForSounds

```
CategoryItem [] OSSC.SoundCueData.categoriesForSounds
```

category items that correspond with each of SoundItem in sounds.

Definition at line 291 of file SoundCue.cs.

2.13.2.3 categoryVolumes

```
float [] OSSC.SoundCueData.categoryVolumes
```

Category sound volumes that correspond with Sound items.

Definition at line 295 of file SoundCue.cs.

2.13.2.4 fadeInTime

```
float OSSC.SoundCueData.fadeInTime
```

Fade In time.

Definition at line 303 of file SoundCue.cs.

2.13.2.5 fadeOutTime

```
float OSSC.SoundCueData.fadeOutTime
```

Fade Out time.

Definition at line 307 of file SoundCue.cs.

2.13.2.6 isFadeIn

```
bool OSSC.SoundCueData.isFadeIn
```

Should SoundCue Fade In?

Definition at line 311 of file SoundCue.cs.

2.13.2.7 isFadeOut

```
bool OSSC.SoundCueData.isFadeOut
```

Should SoundCue Fade Out?

Definition at line 315 of file SoundCue.cs.

2.13.2.8 isLooped

```
bool OSSC.SoundCueData.isLooped
```

Should SoundCue be looped?

Definition at line 320 of file SoundCue.cs.

2.13.2.9 sounds

```
SoundItem [] OSSC.SoundCueData.sounds
```

sound items that played by the SoundCue.

Definition at line 287 of file SoundCue.cs.

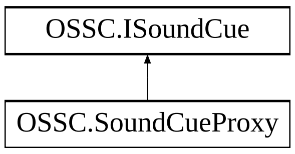
The documentation for this struct was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundCue.cs

2.14 OSSC.SoundCueProxy Class Reference

Given by the SoundController to User as ISoundCue to control the playing SoundCue.

Inheritance diagram for OSSC.SoundCueProxy:



Public Member Functions

- void Play (SoundCueData data)

  Check ISoundCue.
- void Pause ()

  Check ISoundCue.
- void Resume ()

  Check ISoundCue.
- void Stop (bool shouldCallOnFinishedCue=true)

  Check ISoundCue.

Properties

- SoundCue SoundCue `[get, set]`

  Sets, Gets the SoundCue.
- Action< string > OnPlayEnded `[get, set]`

  Check ISoundCue
- Action< SoundCue > OnPlayCueEnded `[get, set]`

  Check ISoundCue
- SoundObject AudioObject `[get, set]`

  Check ISoundCue
- SoundCueData Data `[get]`

  Check ISoundCue
- bool IsPlaying `[get]`

  Check ISoundCue
- int ID `[get]`

  Check ISoundCue

2.14.1 Detailed Description

Given by the SoundController to User as ISoundCue to control the playing SoundCue.

Definition at line 11 of file SoundCueProxy.cs.

2.14.2 Member Function Documentation

2.14.2.1 Pause()

`void OSSC.SoundCueProxy.Pause ( )`

Check ISoundCue.

Implements OSSC.ISoundCue.

Definition at line 127 of file SoundCueProxy.cs.

2.14.2.2 Play()

```
void OSSC.SoundCueProxy.Play (
            SoundCueData data )
```

Check ISoundCue.

Implements OSSC.ISoundCue.

Definition at line 113 of file SoundCueProxy.cs.

2.14.2.3 Resume()

```
void OSSC.SoundCueProxy.Resume ( )
```

Check ISoundCue.

Implements OSSC.ISoundCue.

Definition at line 140 of file SoundCueProxy.cs.

2.14.2.4 Stop()

```
void OSSC.SoundCueProxy.Stop (
            bool shouldCallOnFinishedCue = true )
```

Check ISoundCue.

Implements OSSC.ISoundCue.

Definition at line 153 of file SoundCueProxy.cs.

2.14.3 Property Documentation

2.14.3.1 AudioObject

```
SoundObject OSSC.SoundCueProxy.AudioObject  [get], [set]
```

Check ISoundCue

Definition at line 73 of file SoundCueProxy.cs.

2.14.3.2 Data

```
SoundCueData OSSC.SoundCueProxy.Data  [get]
```

Check ISoundCue

Definition at line 88 of file SoundCueProxy.cs.

2.14.3.3 ID

```
int OSSC.SoundCueProxy.ID  [get]
```

Check ISoundCue

Definition at line 104 of file SoundCueProxy.cs.

2.14.3.4 IsPlaying

```
bool OSSC.SoundCueProxy.IsPlaying  [get]
```

Check ISoundCue

Definition at line 94 of file SoundCueProxy.cs.

2.14.3.5 OnPlayCueEnded

```
Action<SoundCue> OSSC.SoundCueProxy.OnPlayCueEnded  [get], [set]
```

Check ISoundCue

Definition at line 58 of file SoundCueProxy.cs.

2.14.3.6 OnPlayEnded

```
Action<string> OSSC.SoundCueProxy.OnPlayEnded  [get], [set]
```

Check ISoundCue

Definition at line 43 of file SoundCueProxy.cs.

2.14.3.7 SoundCue

```
SoundCue OSSC.SoundCueProxy.SoundCue  [get], [set]
```

Sets, Gets the SoundCue.

Definition at line 28 of file SoundCueProxy.cs.

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

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundCueProxy.cs

2.15 OSSC.Model.SoundItem Class Reference

Used by CategoryItem to store sounds data.

Public Attributes

- string name

  SoundItem Name
- int tagID = -1

  Tag ID associated with SoundItem
- UnityEngine.Audio.AudioMixerGroup mixer

  Mixer group associated with this SoundItem.
- AudioClip [ ] clips

  List of Audioclips
- bool isRandomPitch

  Is SoundItem using Random Pitch?
- CustomRange pitchRange = new CustomRange()

  Range of the Random pitch.
- bool isRandomVolume

  Is SoundItem using Random Volume?
- CustomRange volumeRange = new CustomRange()

  Range of the Random Volume.
- float volume = 1f

  Standard volume of the SoundItem

2.15.1 Detailed Description

Used by CategoryItem to store sounds data.

Definition at line 9 of file SoundItem.cs.

2.15.2 Member Data Documentation

2.15.2.1 clips

`AudioClip [ ] OSSC.Model.SoundItem.clips`

List of Audioclips

Definition at line 26 of file SoundItem.cs.

2.15.2.2 isRandomPitch

`bool OSSC.Model.SoundItem.isRandomPitch`

Is SoundItem using Random Pitch?

Definition at line 30 of file SoundItem.cs.

2.15.2.3 isRandomVolume

`bool OSSC.Model.SoundItem.isRandomVolume`

Is SoundItem using Random Volume?

Definition at line 38 of file SoundItem.cs.

2.15.2.4 mixer

`UnityEngine.Audio.AudioMixerGroup OSSC.Model.SoundItem.mixer`

Mixer group associated with this SoundItem.

Definition at line 22 of file SoundItem.cs.

2.15.2.5 name

`string OSSC.Model.SoundItem.name`

SoundItem Name

Definition at line 14 of file SoundItem.cs.

2.15.2.6 pitchRange

```
CustomRange OSSC.Model.SoundItem.pitchRange = new CustomRange()
```

Range of the Random pitch.

Definition at line 34 of file SoundItem.cs.

2.15.2.7 tagID

```
int OSSC.Model.SoundItem.tagID = -1
```

Tag ID associated with SoundItem

Definition at line 18 of file SoundItem.cs.

2.15.2.8 volume

```
float OSSC.Model.SoundItem.volume = 1f
```

Standard volume of the SoundItem

Definition at line 48 of file SoundItem.cs.

2.15.2.9 volumeRange

```
CustomRange OSSC.Model.SoundItem.volumeRange = new CustomRange()
```

Range of the Random Volume.

Definition at line 42 of file SoundItem.cs.

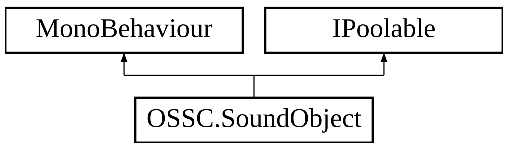
The documentation for this class was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Model/SoundItem.cs

2.16 OSSC.SoundObject Class Reference

Used by the SoundCue. Controls the AudioSource.

Inheritance diagram for OSSC.SoundObject:



Public Member Functions

- void Setup (string id, AudioClip clip, float volume, float fadeInTime=0f, float fadeOutTime=0f, AudioMixer↩ Group mixer=null, float pitch=1f)

  Prepares the SoundObject for playing an AudioClip.
- void Play ()

  Plays the AudioSource.
- void Pause ()

  Pauses the AudioSource.
- void Resume ()

  Resumes from Pause.
- void Stop ()

  Stops the SoundObject from playing.
- bool IsFree ()

  Check IPoolable

Public Attributes

- System.Action< SoundObject > OnFinishedPlaying

  Called when SoundObject finishes playing.

Properties

- bool isDespawnOnFinishedPlaying `[get, set]`

  Check whether SoundObject should despawn after finishing playing.
- string clipName `[get]`

  AudioClip name played.
- AudioSource source `[get]`

  Gets the SoundObject's AudioSource.
- string ID `[get]`

  Gets the SoundObject's ID.

2.16.1 Detailed Description

Used by the SoundCue. Controls the AudioSource.

Definition at line 14 of file SoundObject.cs.

2.16.2 Member Function Documentation

2.16.2.1 IsFree()

```
bool OSSC.SoundObject.IsFree ( )
```

Check IPoolable

Implements IPoolable.

Definition at line 291 of file SoundObject.cs.

2.16.2.2 Pause()

```
void OSSC.SoundObject.Pause ( )
```

Pauses the AudioSource.

Definition at line 153 of file SoundObject.cs.

2.16.2.3 Play()

```
void OSSC.SoundObject.Play ( )
```

Plays the AudioSource.

Definition at line 137 of file SoundObject.cs.

2.16.2.4 Resume()

```
void OSSC.SoundObject.Resume ( )
```

Resumes from Pause.

Definition at line 164 of file SoundObject.cs.

2.16.2.5 Setup()

```
void OSSC.SoundObject.Setup (
             string id,
             AudioClip clip,
             float volume,
             float fadeInTime = 0f,
             float fadeOutTime = 0f,
             AudioMixerGroup mixer = null,
             float pitch = 1f )
```

Prepares the SoundObject for playing an AudioClip.

Parameters

id	SoundObject's ID
clip	AudioClip to play
volume	volume of the sound.
fadeInTime	Fade In Time
fadeOutTime	Fade Out Time
mixer	Audio Mixer group
pitch	Pitch of the sound

Definition at line 117 of file SoundObject.cs.

2.16.2.6 Stop()

```
void OSSC.SoundObject.Stop ( )
```

Stops the SoundObject from playing.

Definition at line 175 of file SoundObject.cs.

2.16.3 Member Data Documentation

2.16.3.1 OnFinishedPlaying

```
System.Action<SoundObject> OSSC.SoundObject.OnFinishedPlaying
```

Called when SoundObject finishes playing.

Definition at line 20 of file SoundObject.cs.

2.16.4 Property Documentation

2.16.4.1 clipName

```
string OSSC.SoundObject.clipName  [get]
```

AudioClip name played.

Definition at line 85 of file SoundObject.cs.

2.16.4.2 ID

```
string OSSC.SoundObject.ID  [get]
```

Gets the SoundObject's ID.

Definition at line 103 of file SoundObject.cs.

2.16.4.3 isDespawnOnFinishedPlaying

```
bool OSSC.SoundObject.isDespawnOnFinishedPlaying  [get], [set]
```

Check whether SoundObject should despawn after finishing playing.

Definition at line 76 of file SoundObject.cs.

2.16.4.4 source

```
AudioSource OSSC.SoundObject.source  [get]
```

Gets the SoundObject's AudioSource.

Definition at line 96 of file SoundObject.cs.

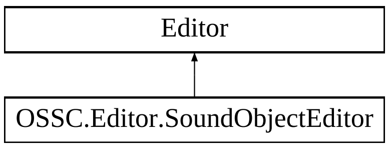
The documentation for this class was generated from the following file:

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundObject.cs

2.17 OSSC.Editor.SoundObjectEditor Class Reference

Draw the custom editor inspector for SoundObject

Inheritance diagram for OSSC.Editor.SoundObjectEditor:



Public Member Functions

- override void OnInspectorGUI ()
  Draws the inspector's GUI

2.17.1 Detailed Description

Draw the custom editor inspector for SoundObject

Definition at line 13 of file SoundObjectEditor.cs.

2.17.2 Member Function Documentation

2.17.2.1 OnInspectorGUI()

```
override void OSSC.Editor.SoundObjectEditor.OnInspectorGUI ( )
```

Draws the inspector's GUI

Definition at line 27 of file SoundObjectEditor.cs.

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

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Editor/SoundObjectEditor.cs

2.18 OSSC.SoundTags Class Reference

Used By the SoundController for tagging SoundItems.

Public Member Functions

- SoundTags ()

  Default Constructor
- TagData [ ] ToArray ()

  Returns all data in form of an array
- string [ ] ToArrayNames ()

  Returns the names of the tags.
- int [ ] ToArrayIDs ()

  Returns the IDs of the tags.
- TagData GetTagDataByName (string name)

  Gets TagData by name.
- TagData GetTagDataByID (int ID)

  Gets TagData by ID.
- int GetTagIDByName (string name)

  Gets Tag ID by name.
- string GetTagNameByID (int ID)

  Gets Tag name by ID
- void SetTag (string name)

  Sets a new Tag.
- void RemoveByTag (TagData data)

  Removes a Tag by TagData.

2.18.1 Detailed Description

Used By the SoundController for tagging SoundItems.

Definition at line 10 of file SoundTags.cs.

2.18.2 Constructor & Destructor Documentation

2.18.2.1 SoundTags()

```
OSSC.SoundTags.SoundTags ( )
```

Default Constructor

Definition at line 25 of file SoundTags.cs.

2.18.3 Member Function Documentation

2.18.3.1 GetTagDataByID()

```
TagData OSSC.SoundTags.GetTagDataByID (
             int ID )
```

Gets TagData by ID.

Parameters

ID	ID of the Tag.

Returns

TagData with the corresponding ID.

Definition at line 86 of file SoundTags.cs.

2.18.3.2 GetTagDataByName()

```
TagData OSSC.SoundTags.GetTagDataByName (
             string name )
```

Gets TagData by name.

Parameters

name	name of the Tag.

Returns

TagData with the corresponding name.

Definition at line 76 of file SoundTags.cs.

2.18.3.3 GetTagIDByName()

```
int OSSC.SoundTags.GetTagIDByName (
            string name )
```

Gets Tag ID by name.

Parameters

name	name of the Tag.

Returns

ID of the Tag.

Definition at line 96 of file SoundTags.cs.

2.18.3.4 GetTagNameByID()

```
string OSSC.SoundTags.GetTagNameByID (
            int ID )
```

Gets Tag name by ID

Parameters

ID	ID of the Tag

Returns

Name of the Tag.

Definition at line 110 of file SoundTags.cs.

2.18.3.5 RemoveByTag()

```
void OSSC.SoundTags.RemoveByTag (
             TagData data )
```

Removes a Tag by TagData.

Parameters

data	TagData that wants to be removed.

Definition at line 143 of file SoundTags.cs.

2.18.3.6 SetTag()

```
void OSSC.SoundTags.SetTag (
             string name )
```

Sets a new Tag.

Parameters

name	Name of the Tag.

Definition at line 123 of file SoundTags.cs.

2.18.3.7 ToArray()

```
TagData [] OSSC.SoundTags.ToArray ( )
```

Returns all data in form of an array

Returns

Array of TagData.

Definition at line 36 of file SoundTags.cs.

2.18.3.8 ToArrayIDs()

```
int [] OSSC.SoundTags.ToArrayIDs ( )
```

Returns the IDs of the tags.

Returns

int[] with ids

Definition at line 60 of file SoundTags.cs.

2.18.3.9 ToArrayNames()

```
string [ ] OSSC.SoundTags.ToArrayNames ( )
```

Returns the names of the tags.

Returns

string[] with names

Definition at line 45 of file SoundTags.cs.

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

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundTags.cs

2.19 OSSC.TagData Struct Reference

Used by the SoundTags to save Tags.

Public Attributes

- string name
  Tag Name
- int ID
  Tag ID

2.19.1 Detailed Description

Used by the SoundTags to save Tags.

Definition at line 154 of file SoundTags.cs.

2.19.2 Member Data Documentation

2.19.2.1 ID

```
int OSSC.TagData.ID
```

Tag ID

Definition at line 163 of file SoundTags.cs.

2.19.2.2 name

```
string OSSC.TagData.name
```

Tag Name

Definition at line 159 of file SoundTags.cs.

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

- F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundTags.cs

Chapter 3

File Documentation

3.1 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Cue↩ Manager.cs File Reference

Classes

- class OSSC.CueManager

  Manages all SoundCues

Namespaces

- namespace OSSC

3.2 CueManager.cs

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

namespace OSSC
{
    public class CueManager
    {
        #region Private fields
        private List<SoundCue> _soundCues;
        #endregion

        #region Public Methods and Properties
        public CueManager()
        {
            _soundCues = new List<SoundCue>();
        }

        public CueManager(int initialSize)
        {
            _soundCues = new List<SoundCue>(initialSize);
        }

        public SoundCue GetSoundCue()
        {
            SoundCue cue = FindFreeCue();
            cue.OnPlayKilled += OnPlayKilled_handler;
            return cue;
        }

        public void StopAllCues(bool shouldCallOnEndCallback = true)
        {
            for (int i = 0; i < _soundCues.Count; i++)
            {
                if (_soundCues[i].IsPlaying)
                    _soundCues[i].Stop(shouldCallOnEndCallback);
            }
        }
        #endregion

        #region Private methods
        private void OnPlayKilled_handler(SoundCue cue, SoundCueProxy proxy)
        {
            //NOTE: Clear up any references to events.
            cue.OnPlayKilled = null;
            cue.OnPlayCueEnded = null;
            cue.OnPlayEnded = null;
            proxy.SoundCue = null;
        }

        private SoundCue FindFreeCue()
        {
            SoundCue cue = null;
            for (int i = 0; i < _soundCues.Count; i++)
            {
                if (_soundCues[i].IsPlaying == false)
                {
                    cue = _soundCues[i];
                    break;
                }
            }

            if (cue == null)
            {
                cue = new SoundCue(_soundCues.Count);
                _soundCues.Add(cue);
            }

            return cue;
        }
        #endregion
    }
}
```

3.3 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Editor/↩ SoundControllerEditor.cs File Reference

Classes

- class OSSC.Editor.SoundControllerEditor

  Draws the Custom Editor for SoundController

Namespaces

- namespace OSSC.Editor

3.4 SoundControllerEditor.cs

```
using System.Collections;
using System.Collections.Generic;
using System.Security.Cryptography;
using UnityEngine;
using UnityEditor;
using OSSC.Model;
using UnityEngine.Audio;
using UnityEngine.EventSystems;

namespace OSSC.Editor
{
    [CustomEditor(typeof(SoundController))]
    public class SoundControllerEditor : UnityEditor.Editor
    {
        private const int NAME_ABV_LEN = 50;
        private const float PITCH_RANGE_MAX = 3f;
        private const float PITCH_RANGE_MIN = -3f;
        private SoundController _ac;
        private string categoryNameSearch = "";
        private string _tagName = "";

        public override void OnInspectorGUI()
        {
            base.OnInspectorGUI();
            _ac = target as SoundController;

            if (_ac._database == null)
            {
                EditorGUILayout.HelpBox("Create SoundControllerData asset, then throw it here.", MessageType.Info);
            }
            else
            {
                DrawMain();
            }

            EditorUtility.SetDirty(_ac);
            if (_ac._database != null)
                EditorUtility.SetDirty(_ac._database);
        }

        private void DrawMain()
        {
            if (_ac._database == null)
                return;
            DrawSoundTags();
            EditorGUILayout.BeginHorizontal(EditorStyles.helpBox);
            if (GUILayout.Button("DELETE DATA"))
            {
                AssetDatabase.DeleteAsset(AssetDatabase.GetAssetPath(_ac._database));
                return;
            }
            var db = _ac._database;
            if (GUILayout.Button("ADD CATEGORY"))
            {
                var category = new Model.CategoryItem();
                var categories = new Model.CategoryItem[db.items != null ? db.items.Length + 1 : 1];
                if (db.items != null)
                    db.items.CopyTo(categories, 0);
                categories[categories.Length - 1] = category;
                db.items = categories;
            }
            EditorGUILayout.EndHorizontal();
            DrawCategories(_ac._database);
        }

        private void DrawCategories(Model.SoundControllerData db)
        {

            if (db.items == null)
                return;
            if (db.items.Length == 0)
                return;

            categoryNameSearch = EditorGUILayout.TextField("Search Category", categoryNameSearch);
            db.foldOutCategories = EditorGUILayout.Foldout(db.foldOutCategories, "CATEGORIES", true);
            if (!db.foldOutCategories)
                return;

            for (int i = db.items.Length - 1; i >= 0; i--)
            {
                if (!string.IsNullOrEmpty(db.items[i].name))
                    if (db.items[i].name.ToLower().Contains(categoryNameSearch.ToLower()) == false && string.IsNullOrEmpty(categoryNameSearch) == false)
                        continue;
                DrawCategory(db.items[i], i);
            }
        }

        private void DrawCategory(Model.CategoryItem item, int index)
        {
            EditorGUILayout.BeginVertical(EditorStyles.helpBox);

            item.name = EditorGUILayout.TextField("Name", item.name);
            item.audioObjectPrefab = (GameObject)EditorGUILayout.ObjectField("Category AO prefab", item.audioObjectPrefab, typeof(GameObject), false);
            item.usingDefaultPrefab = item.audioObjectPrefab == null;
            item.isMute = EditorGUILayout.Toggle("Is Mute", item.isMute);
            item.categoryVolume = EditorGUILayout.Slider("Category Volume", item.categoryVolume, 0f, 1f);

            EditorGUILayout.BeginHorizontal(EditorStyles.helpBox);
            if (GUILayout.Button("ADD SOUND ITEM"))
            {
                var soundItem = new Model.SoundItem();
                bool isNoSoundItems = item.soundItems == null;
                var soundItems = new Model.SoundItem[!isNoSoundItems ? item.soundItems.Length + 1 : 1];
                if (!isNoSoundItems)
                    item.soundItems.CopyTo(soundItems, 0);
                soundItems[soundItems.Length - 1] = soundItem;
                item.soundItems = soundItems;
            }
            string nameAbv = "";
            if (string.IsNullOrEmpty(item.name) == false)
                nameAbv = item.name.Length > NAME_ABV_LEN ? item.name.Substring(0, NAME_ABV_LEN) : item.
      name;
            if (GUILayout.Button("Delete " + nameAbv))
            {
                DeleteCategory(index);
            }
            EditorGUILayout.EndHorizontal();

            if (item.soundItems != null)
                if (item.soundItems.Length != 0)
                    item.soundsSearchName = EditorGUILayout.TextField("Search sound item", item.
      soundsSearchName);

            item.foldOutSoundItems = DrawSoundItems(item, item.foldOutSoundItems, item.soundsSearchName);

            EditorGUILayout.EndVertical();
        }

        private void DeleteCategory(int index)
        {
            var categories = new Model.CategoryItem[_ac._database.items.Length - 1];
            int catInd = 0;
            for (int i = 0; i < _ac._database.items.Length; i++)
            {
                if (i == index)
                    continue;

                categories[catInd] = _ac._database.items[i];
                catInd += 1;
            }
            _ac._database.items = categories;
        }

        private bool DrawSoundItems(Model.CategoryItem item, bool foldOut, string searchName)
        {
            Model.SoundItem[] items = item.soundItems;
            if (items == null || items.Length == 0)
                return foldOut;

            EditorGUI.indentLevel++;

            EditorGUILayout.BeginHorizontal();
            foldOut = EditorGUILayout.Foldout(foldOut, "SOUND ITEMS", true);
            if (items != null)
            {
                if (items.Length != 0)
                    if (GUILayout.Button("DELETE ALL SOUNDS"))
                    {
                        items = new Model.SoundItem[0];
                        item.soundItems = items;
                        return foldOut;
                    }
            }
            EditorGUILayout.EndHorizontal();

            if (foldOut)
            {
                for (int j = items.Length - 1; j >= 0; j--)
                {
                    if (!string.IsNullOrEmpty(items[j].name))
                        if (items[j].name.ToLower().Contains(searchName.ToLower()) == false && string.
      IsNullOrEmpty(searchName) == false)
                            continue;
                    DrawSoundItem(items[j], j, items);
                }
            }
            EditorGUI.indentLevel--;
            return foldOut;
        }

        private void DrawSoundItem(Model.SoundItem item, int index, Model.SoundItem[] items)
        {
            EditorGUILayout.BeginVertical(EditorStyles.helpBox);
            item.name = EditorGUILayout.TextField("Name", item.name);

            string[] names = _ac._database.soundTags.ToArrayNames();
            int[] ids = _ac._database.soundTags.ToArrayIDs();
            if (ids.Length != 0)
            {
                int indexTag = System.Array.IndexOf(ids, item.tagID);
                indexTag = EditorGUILayout.Popup("Tag", indexTag, names);
                if (indexTag != -1)
                    item.tagID = ids[indexTag];
            }
            else
            {
                item.tagID = -1;
            }


            item.mixer = (AudioMixerGroup)EditorGUILayout.ObjectField("Mixer", item.mixer, typeof(AudioMixerGroup), false);
            if (item.clips == null)
            {
                item.clips = new AudioClip[1];
            }
            int size = item.clips.Length;
            size = EditorGUILayout.IntField("Size", size);
            if (size != item.clips.Length)
            {
                var newClips = new AudioClip[size];
                for (int i = 0; i < item.clips.Length; i++)
                {
                    if (i >= size)
                        break;
                    newClips[i] = item.clips[i];
                }
                item.clips = newClips;
            }
            for (int i = 0; i < item.clips.Length; i++)
            {
                item.clips[i] = (AudioClip)EditorGUILayout.ObjectField(item.clips[i], typeof(AudioClip), false);
            }

            item.isRandomVolume =
                EditorGUILayout.ToggleLeft("Use Random Volume", item.isRandomVolume, EditorStyles.boldLabel);
            if (!item.isRandomVolume)
                item.volume = EditorGUILayout.Slider("Volume", item.volume, 0f, 1f);
            else
            {
                EditorGUILayout.LabelField("Min Volume:", item.volumeRange.min.ToString(), EditorStyles.largeLabel);
                EditorGUILayout.LabelField("Max Volume:", item.volumeRange.max.ToString(), EditorStyles.largeLabel);
                EditorGUILayout.MinMaxSlider("Volume Range", ref item.volumeRange.min, ref item.volumeRange.max, 0f, 1f);
            }

            item.isRandomPitch =
                EditorGUILayout.ToggleLeft("Use Random Pitch", item.isRandomPitch, EditorStyles.boldLabel);
            if (item.isRandomPitch)
            {
                EditorGUILayout.LabelField("Min Pitch:", item.pitchRange.min.ToString(), EditorStyles.largeLabel);
                EditorGUILayout.LabelField("Max Pitch:", item.pitchRange.max.ToString(), EditorStyles.largeLabel);
                EditorGUILayout.MinMaxSlider("Pitch Range", ref item.pitchRange.min, ref item.pitchRange.max, PITCH_RANGE_MIN, PITCH_RANGE_MAX);
            }
            string nameAbv = "";
            if (string.IsNullOrEmpty(item.name) == false)
                nameAbv = item.name.Length > NAME_ABV_LEN ? item.name.Substring(0, NAME_ABV_LEN) : item.name;
            if (GUILayout.Button("Delete Item " + nameAbv))
            {
                DeleteSoundItem(index, items);
            }
            EditorGUILayout.EndVertical();
        }

        private void DeleteSoundItem(int index, Model.SoundItem[] items)
        {
            var category = System.Array.Find(_ac._database.items, (x) => {
                return x.soundItems == items;
            });

            var soundItems = new Model.SoundItem[category.soundItems.Length - 1];
            int soundInd = 0;
            for (int i = 0; i < category.soundItems.Length; i++)
            {
                if (i == index)
                    continue;
                soundItems[soundInd] = category.soundItems[i];
                soundInd += 1;
            }
            category.soundItems = soundItems;
        }

        private void DrawSoundTags()
        {
            if (_ac._database.soundTags == null)
            {
                _ac._database.soundTags = new SoundTags();
            }

            _ac._database.foldOutTags = EditorGUILayout.Foldout(_ac.
      _database.foldOutTags, "Tags", true);
            if (_ac._database.foldOutTags == false)
            {
                EditorGUILayout.HelpBox("Add tags filter sounds by them.", MessageType.Info);
                return;
            }

            EditorGUILayout.BeginVertical(EditorStyles.helpBox);
            DrawAddNewTag();

            TagData[] data = _ac._database.soundTags.
      ToArray();
            for (int i = 0; i < data.Length; i++)
            {
                DrawSoundTag(data[i], _ac._database.soundTags);
            }
            EditorGUILayout.EndVertical();
        }

        private void DrawSoundTag(TagData data, SoundTags tags)
        {
            EditorGUILayout.BeginHorizontal();
            EditorGUILayout.LabelField("ID: " + data.ID.ToString(), "name: " + data.
      name);
            if (GUILayout.Button("Delete"))
            {
                tags.RemoveByTag(data);
            }
            EditorGUILayout.EndHorizontal();
        }

        private void DrawAddNewTag()
        {
            EditorGUILayout.BeginHorizontal(EditorStyles.helpBox);
            _tagName = EditorGUILayout.TextField("Add Tag:", _tagName);
            if (GUILayout.Button("Add"))
            {
                _ac._database.soundTags.SetTag(_tagName);
                _tagName = string.Empty;
            }
            EditorGUILayout.EndHorizontal();
        }
    }
}
```

3.5 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Editor/↩ SoundObjectEditor.cs File Reference

Classes

- class OSSC.Editor.SoundObjectEditor

  Draw the custom editor inspector for SoundObject

Namespaces

- namespace OSSC.Editor

3.6 SoundObjectEditor.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEditor;
using OSSC;

namespace OSSC.Editor
{
    [CustomEditor(typeof(SoundObject))]
    public class SoundObjectEditor : UnityEditor.Editor
    {
        private SoundObject _ao;
        private bool _showControls = false;

        public override void OnInspectorGUI()
        {
            _ao = target as SoundObject;

            EditorGUILayout.LabelField("Sound Item", _ao.ID, EditorStyles.boldLabel);
            EditorGUILayout.LabelField("Current Clip", _ao.clipName);
            _showControls = EditorGUILayout.ToggleLeft("Show Controls", _showControls);
            if (_showControls == false)
                return;

            ShowControls();
        }

        private void ShowControls()
        {
            if (_ao.source.isPlaying)
            {
                if (GUILayout.Button("Stop"))
                {
                    _ao.Stop();
                }
            }

            if (_ao.source.clip != null)
            {
                int minutes = (int)(_ao.source.time / 60f);
                int seconds = (int)(_ao.source.time - minutes * 60);
                EditorGUILayout.LabelField("Current Time", minutes + ":" + seconds);
                minutes = (int)(_ao.source.clip.length / 60f);
                seconds = (int)(_ao.source.clip.length - minutes * 60);
                EditorGUILayout.LabelField("Clip Time", minutes + ":" + seconds);
                _ao.source.time = EditorGUILayout.Slider("Seek", _ao.
source.time, 0f, _ao.source.clip.length);
                Repaint();
            }
        }
    }
}
```

3.7 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/IPoolable.cs File Reference

Classes

- interface IPoolable

  Used by the ObjectPool

3.8 IPoolable.cs

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

public interface IPoolable
{
    PrefabBasedPool pool {
        get; set;
    }

    bool IsFree();
}
```

3.9 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/ISound↩ Cue.cs File Reference

Classes

- interface OSSC.ISoundCue

  SoundCue Inteface. SoundController returns a SoundCue Interface to further control the playing SouncCue

Namespaces

- namespace OSSC

3.10 ISoundCue.cs

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

namespace OSSC
{
    public interface ISoundCue
    {
        Action<string> OnPlayEnded { get; set; }
        Action<SoundCue> OnPlayCueEnded { get; set; }
        SoundObject AudioObject { get; set; }
        SoundCueData Data { get; }
        bool IsPlaying { get; }
        int ID { get; }

        void Play(SoundCueData data);
        void Pause();
        void Resume();
        void Stop(bool shouldCallOnFinishedCue = true);
    }
}
```

3.11 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Model/↩ CategoryItem.cs File Reference

Classes

- class OSSC.Model.CategoryItem

  Used by the SoundControllerData to store categories.

Namespaces

- namespace OSSC.Model

3.12 CategoryItem.cs

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

namespace OSSC.Model
{
    [System.Serializable]
    public class CategoryItem
    {
        public string name;
        public SoundItem[] soundItems;
        public GameObject audioObjectPrefab;
        public bool usingDefaultPrefab = true;

        [Range(0f, 1f)]
        public float categoryVolume = 1f;

        public bool foldOutSoundItems = false;
        public string soundsSearchName = "";
        public bool isMute = false;
    }

}
```

3.13 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Model/↩ SoundControllerData.cs File Reference

Classes

- class OSSC.Model.SoundControllerData

  SoundController's Database.

Namespaces

- namespace OSSC.Model

3.14 SoundControllerData.cs

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

namespace OSSC.Model
{
    [CreateAssetMenu(fileName = "NewSoundControllerData", menuName = "Sound Controller/New
  SoundControllerData")]
    public class SoundControllerData : ScriptableObject
    {
        public CategoryItem[] items;
        public bool foldOutCategories = false;
        public bool foldOutTags = false;
        public string assetName;
        public SoundTags soundTags;
    }
}
```

3.15 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Model/↩ SoundItem.cs File Reference

Classes

- class OSSC.Model.SoundItem

  Used by CategoryItem to store sounds data.
- class OSSC.Model.CustomRange

  Used by SoundItem to store Random Ranges.

Namespaces

- namespace OSSC.Model

3.16 SoundItem.cs

```
using UnityEngine;

namespace OSSC.Model
{
    [System.Serializable]
    public class SoundItem
    {
        public string name;
        public int tagID = -1;
        public UnityEngine.Audio.AudioMixerGroup mixer;
        public AudioClip[] clips;
        public bool isRandomPitch;
        public CustomRange pitchRange = new CustomRange();
        public bool isRandomVolume;
        public CustomRange volumeRange = new CustomRange();

        [RangeAttribute(0f, 1f)]
        public float volume = 1f;
    }

    [System.Serializable]
    public class CustomRange
    {
        public float min = 1f;
        public float max = 1f;

        public float GetRandomRange()
        {
            return Random.Range(min, max);
        }
    }
}
```

3.17 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Object↩ Pool.cs File Reference

Classes

- class ObjectPool

  Creates a pool of different prefabs when someone requests a GameObject.
- class PrefabBasedPool

3.18 ObjectPool.cs

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

public class ObjectPool : MonoBehaviour
{
    #region Public fields
    public List<PrefabBasedPool> pools;
    #endregion
    #region Public methods and properties

    public GameObject GetFreeObject(GameObject prefab = null)
    {
        if (prefab == null)
            return null;

        PrefabBasedPool pool = pools.Find((x) => {
            return x.prefab == prefab;
        });

        if (pool != null)
            return pool.GetFreeObject();

        pool = new PrefabBasedPool(prefab);
        GameObject parent = new GameObject();
        parent.name = pool.prefab.name + " ::: POOL";
        parent.transform.parent = this.gameObject.transform;
        pool.parent = parent.transform;
        pools.Add(pool);
        return pool.GetFreeObject();
    }

    #endregion

    #region Monobehaviour methods
    void Awake()
    {
        pools = new List<PrefabBasedPool>();
    }
    #endregion
}

[System.Serializable]
public class PrefabBasedPool
{
    public PrefabBasedPool(GameObject prefab)
    {
        pool = new List<GameObject>();
        this.prefab = prefab;
    }
    public GameObject prefab;
    public List<GameObject> pool;

    public Transform parent;

    public GameObject GetFreeObject()
    {
        GameObject freeObj = pool.Find((x) => {
            if (x == null)
            {
                pool.Remove(x);
                return false;
            }
            var poolable = x.GetComponent<IPoolable>();
            return poolable.IsFree();
        });

        if (freeObj != null)
        {
            freeObj.SetActive(true);
            return freeObj;
        }

        var obj = GameObject.Instantiate(prefab, Vector3.zero, Quaternion.identity, parent);
        obj.SetActive(true);
        var objPoolable = obj.GetComponent<IPoolable>();
        objPoolable.pool = this;
        pool.Add(obj);

        return obj;
    }

    public void Despawn(GameObject obj)
    {
        obj.transform.SetParent(parent, false);
        obj.SetActive(false);
    }
}
```

3.19 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Sound↩ Controller.cs File Reference

Classes

- class OSSC.SoundController

    The main class that is used for Playing and controlling all sounds.
- struct OSSC.PlaySoundSettings

    Set the settings to play a particular cue with particular preferences.

Namespaces

- namespace OSSC

3.20 SoundController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using OSSC.Model;

namespace OSSC
{
    [RequireComponent(typeof(ObjectPool))]
    public class SoundController : MonoBehaviour
    {
        #region Serialized Data
        public GameObject _defaultPrefab;
        public SoundControllerData _database;

        #endregion

        #region Private fields

        private ObjectPool _pool;
        private CueManager _cueManager;
        private int _initialCueManagerSize = 10;

        #endregion

        #region Public methods and properties

        public GameObject defaultPrefab
        {
            set
            {
                _defaultPrefab = value;
            }
        }

        public void StopAll(bool shouldCallOnEndCallback = true)
        {
            _cueManager.StopAllCues(shouldCallOnEndCallback);
        }

        public void SetMute(string categoryName, bool value)
        {
            for (int i = 0; i < _database.items.Length; i++)
            {
                if (_database.items[i].name == categoryName)
                {
                    _database.items[i].isMute = value;
                }
            }
        }

        public ISoundCue Play(PlaySoundSettings settings)
        {
            if (settings.soundCueProxy != null)
            {
                return PlaySoundCue(settings);
            }

            if (settings.names == null && string.IsNullOrEmpty(settings.name))
            {
                return null;
            }

            string[] names = null;
            string categoryName = settings.categoryName;
            float fadeInTime = settings.fadeInTime;
            float fadeOutTime = settings.fadeOutTime;
            bool isLooped = settings.isLooped;
            int tagID = _database.soundTags.GetTagIDByName(settings.tagName);
            Transform parent = settings.parent;

            if (settings.names != null)
            {
                names = settings.names;
            }
            else
            {
                names = new[] { settings.name };
            }
            UnityEngine.Assertions.Assert.IsNotNull(names, "[AudioController] names cannot be null");
            if (names != null)
                UnityEngine.Assertions.Assert.IsFalse(names.Length == 0, "[AudioController] names cannot have 0 strings");

            CategoryItem category = null;
            GameObject prefab = null;
            List<SoundItem> items = new List<SoundItem>();
            List<float> catVolumes = new List<float>();
            List<CategoryItem> categories = new List<CategoryItem>();

            if (string.IsNullOrEmpty(categoryName) == false)
            {
                category = System.Array.Find(_database.items, (item) =>
                {
                    return item.name == categoryName;
                });

                // Debug.Log(category);
                if (category == null)
                    return null;

                prefab = category.usingDefaultPrefab ? _defaultPrefab : category.audioObjectPrefab;
                for (int i = 0; i < names.Length; i++)
                {
                    SoundItem item = System.Array.Find(category.soundItems, (x) =>
                    {
                        return x.name == names[i];
                    });

                    if (item != null && category.isMute == false)
                    {
                        bool canAddItem = tagID == -1 || tagID == item.tagID;
                        if (canAddItem)
                        {
                            catVolumes.Add(category.categoryVolume);
                            items.Add(item);
                            categories.Add(category);
                        }
                    }
                }
            }
            else
            {
                prefab = _defaultPrefab;
                CategoryItem[] categoryItems = _database.items;
                for (int i = 0; i < names.Length; i++)
                {
                    SoundItem item = null;
                    item = items.Find((x) => names[i] == x.name);
                    if (item != null)
                    {
                        bool canAddItem = tagID == -1 || tagID == item.tagID;
                        if (canAddItem == false)
                            continue;

                        catVolumes.Add(catVolumes[items.IndexOf(item)]);
                        categories.Add(categories[items.IndexOf(item)]);
                        items.Add(item);
                        continue;
                    }

                    for (int j = 0; j < categoryItems.Length; j++)
                    {
                        item = System.Array.Find(categoryItems[j].soundItems, (x) => x.name == names[i]);
                        if (item != null && categoryItems[j].isMute == false)
                        {
                            bool canAddItem = tagID == -1 || tagID == item.tagID;
                            if (canAddItem == false)
                                continue;
                            catVolumes.Add(categoryItems[j].categoryVolume);
                            categories.Add(categoryItems[j]);
                            items.Add(item);
                            break;
                        }
                    }
                }
            }

            if (items.Count == 0)
                return null;

            SoundCue cue = _cueManager.GetSoundCue();
            SoundCueData data;
            data.audioPrefab = prefab;
            data.sounds = items.ToArray();
            data.categoryVolumes = catVolumes.ToArray();
            data.categoriesForSounds = categories.ToArray();
            data.fadeInTime = fadeInTime;
            data.fadeOutTime = fadeOutTime;
            data.isFadeIn = data.fadeInTime >= 0.1f;
            data.isFadeOut = data.fadeOutTime >= 0.1f;
            data.isLooped = isLooped;
            cue.AudioObject = _pool.GetFreeObject(prefab).GetComponent<SoundObject>();
            if (parent != null)
                cue.AudioObject.transform.SetParent(parent, false);

            SoundCueProxy proxy = new SoundCueProxy();
            proxy.SoundCue = cue;
            proxy.Play(data);
            return proxy;
        }

        #endregion

        #region Private methods
        private SoundCueProxy PlaySoundCue(PlaySoundSettings settings)
        {
            SoundCueProxy cue = settings.soundCueProxy as SoundCueProxy;
            Transform parent = settings.parent;
            float fadeInTime = settings.fadeInTime;
            float fadeOutTime = settings.fadeOutTime;
            bool isLooped = settings.isLooped;
            var ncue = _cueManager.GetSoundCue();
            ncue.AudioObject = _pool.GetFreeObject(cue.Data.audioPrefab).GetComponent<SoundObject>();
            if (parent != null)
                ncue.AudioObject.transform.SetParent(parent, false);
            SoundCueData data = cue.Data;
            data.fadeInTime = fadeInTime;
            data.fadeOutTime = fadeOutTime;
            data.isFadeIn = data.fadeInTime >= 0.1f;
            data.isFadeOut = data.fadeOutTime >= 0.1f;
            data.isLooped = isLooped;
            cue.SoundCue = ncue;
            cue.Play(data);
            return cue;
        }

        #region Internal tests
        [ContextMenu("Test play")]
        void Test()
        {
            PlaySoundSettings settings = new PlaySoundSettings();
            settings.Init();
            settings.name = "Test";
            var proxyCue = Play(settings);
            Debug.Log(proxyCue.ID);
        }

        [ContextMenu("Test Play looped")]
        void TestLoop()
        {
            PlaySoundSettings settings = new PlaySoundSettings();
            settings.Init();
            settings.name = "Test";
            settings.isLooped = true;
            var proxyCue = Play(settings);
            Debug.Log(proxyCue.ID);
        }

        [ContextMenu("Test sequence")]
        void TestSequence()
        {
            PlaySoundSettings settings = new PlaySoundSettings();
            settings.Init();
            settings.names = new[] {"Test", "Test1", "Test2"};
            var proxyCue = Play(settings);
            Debug.Log(proxyCue.ID);
        }

        [ContextMenu("Test sequence looped")]
        void TestSequenceLooped()
        {
            PlaySoundSettings settings = new PlaySoundSettings();
            settings.Init();
            settings.names = new[] {"Test", "Test1", "Test2"};
            settings.isLooped = true;
            var proxyCue = Play(settings);
            Debug.Log(proxyCue.ID);
        }

        [ContextMenu("Test sequence plays 2 times")]
        void TestSequence2TimesPlay()
        {
            PlaySoundSettings settings = new PlaySoundSettings();
            settings.Init();
            settings.names = new[] {"Test", "Test1", "Test2"};
            var proxyCue = Play(settings);
            proxyCue.OnPlayCueEnded += cue =>
            {
                var sett = new PlaySoundSettings();
                sett.soundCueProxy = proxyCue;
                proxyCue = Play(sett);
            };
            Debug.Log(proxyCue.ID);
        }
        #endregion

        #endregion

        #region MonoBehaviour methods

        void Awake()
        {
            _pool = GetComponent<ObjectPool>();
            _cueManager = new CueManager(_initialCueManagerSize);
        }

        #endregion
    }

    public struct PlaySoundSettings
    {
        public string name;
        public string[] names;
        public Transform parent;
        public float fadeInTime;
        public float fadeOutTime;
        public string categoryName;
        public bool isLooped;
        public ISoundCue soundCueProxy;
        public string tagName;

        public void Init()
        {
            name = string.Empty;
            names = null;
            parent = null;
            fadeInTime = 0f;
            fadeOutTime = 0f;
            categoryName = string.Empty;
            isLooped = false;
            soundCueProxy = null;
            tagName = string.Empty;
        }
    }

}
```

3.21 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/SoundCue.cs File Reference

Classes

- class OSSC.SoundCue

  Plays a whole cue of soundItems
- struct OSSC.SoundCueData

  Used for sending data to play to AudioCue

Namespaces

- namespace OSSC

3.22 SoundCue.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;
using OSSC.Model;

namespace OSSC
{
    public class SoundCue : ISoundCue
    {
        public Action<string> OnPlayEnded { get; set; }

        public Action<SoundCue> OnPlayCueEnded { get; set; }

        public Action<SoundCue, SoundCueProxy> OnPlayKilled { get; set; }

        public SoundObject AudioObject { get; set; }

        public SoundCueData Data { get { return _data; } }

        public bool IsPlaying
        {
            get;
            private set;
        }

        public int ID
        {
            get;
            private set;
        }

        public SoundCue()
        {
        }

        public SoundCue(int id)
        {
            ID = id;
        }

        private int _currentItem = 0;
        private SoundCueData _data;
        private SoundCueProxy _currentProxy;

        public void Play(SoundCueData data)
        {
            _data = data;
            AudioObject.isDespawnOnFinishedPlaying = !data.isLooped;
            AudioObject.OnFinishedPlaying = OnFinishedPlaying_handler;
            // audioObject.isDespawnOnFinishedPlaying = false;
            if (TryPlayNext() == false)
            {
                return;
            }
            IsPlaying = true;
        }

        public void Play(SoundCueData data, SoundCueProxy proxy)
        {
            Play(data);
            _currentProxy = proxy;
        }

        public void Pause()
        {
            UnityEngine.Assertions.Assert.IsTrue(_currentItem > 0, "[AudioCue] Cannot pause when not even started.");
            AudioObject.Pause();
        }

        public void Resume()
        {
            UnityEngine.Assertions.Assert.IsTrue(_currentItem > 0, "[AudioCue] Cannot resume when not even started.");
            AudioObject.Resume();
        }

        public void Stop(bool shouldCallOnFinishedCue = true)
        {
            if (IsPlaying == false)
                return;
            AudioObject.OnFinishedPlaying = null;
            // ((IPoolable)audioObject).pool.Despawn(audioObject.gameObject);
            AudioObject.Stop();
            AudioObject = null;
            _currentItem = 0;
            IsPlaying = false;

            if (shouldCallOnFinishedCue)
            {
                if (OnPlayCueEnded != null)
                {
                    OnPlayCueEnded(this);
                }
            }

            if (OnPlayKilled != null)
            {
                OnPlayKilled(this, _currentProxy);
                _currentProxy = null;
            }
        }

        private void OnFinishedPlaying_handler(SoundObject obj)
        {
            string itemName = _data.sounds[_currentItem - 1].name;
            if (OnPlayEnded != null) {
                OnPlayEnded(itemName);
            }

            if (_currentItem < _data.sounds.Length)
            {
                if (TryPlayNext() == false)
                {
                    Stop(true);
                }
            }
            else
            {
                if (_data.isLooped)
                {
                    _currentItem = 0;
                    if (TryPlayNext() == false)
                    {
                        Stop(true);
                    }
                }
                else
                {
                    Stop(true);
                }
            }
        }

        private bool TryPlayNext()
        {
            bool isPlaying = false;
            if (_data.categoriesForSounds[_currentItem].
      isMute == false)
            {
                PlayCurrentItem();
                _currentItem += 1;
                isPlaying = true;
            }
            else
            {
                for (int i = _currentItem; i < _data.sounds.Length; i++)
                {
                    if (_data.categoriesForSounds[i].isMute == false)
                    {
                        _currentItem = i;
                        PlayCurrentItem();
                        _currentItem += 1;
                        isPlaying = true;
                        break;
                    }
                }
            }
            return isPlaying;
        }

        private void PlayCurrentItem()
        {
            SoundItem item = _data.sounds[_currentItem];

            float itemVolume = item.isRandomVolume
                ? item.volumeRange.GetRandomRange()
                : item.volume;
            float realVolume = itemVolume * _data.categoryVolumes[_currentItem];

            float realPitch = item.isRandomPitch
                ? item.pitchRange.GetRandomRange()
                : 1f;

            if (_currentItem == _data.sounds.Length - 1)
            {
                AudioObject.Setup(
                    item.name,
                    GetRandomClip( item.clips ),
                    realVolume,
                    _data.fadeInTime,
                    _data.fadeOutTime,
                    item.mixer,
                    realPitch);
            }
            else
            {
                AudioObject.Setup(
                    item.name,
                    GetRandomClip( item.clips ),
                    realVolume,
                    mixer: item.mixer,
                    pitch: realPitch);
            }
            AudioObject.Play();
        }

        private AudioClip GetRandomClip(AudioClip[] clips)
        {
            int index = UnityEngine.Random.Range(0, clips.Length);
            return clips[index];
        }
    }

    public struct SoundCueData
    {
        public SoundItem[] sounds;
        public CategoryItem[] categoriesForSounds;
        public float[] categoryVolumes;
        public GameObject audioPrefab;
        public float fadeInTime;
        public float fadeOutTime;
        public bool isFadeIn;
        public bool isFadeOut;
        public bool isLooped;
    }
}
```

3.23 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Sound↩ CueProxy.cs File Reference

Classes

- class OSSC.SoundCueProxy

    Given by the SoundController to User as ISoundCue to control the playing SoundCue.

Namespaces

- namespace OSSC

3.24 SoundCueProxy.cs

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

namespace OSSC
{
    public class SoundCueProxy : ISoundCue
    {
        #region Private fields
        private SoundCue _soundCue;
        private SoundCueData _data;
        #endregion
        #region Public Methods and Properties
        public SoundCue SoundCue
        {
            get {
                return _soundCue;
            }
            set {
                _soundCue = value;
            }
        }
        #endregion

        #region ISoundCue implementation
        public Action<string> OnPlayEnded
        {
            get {
                return _soundCue == null ? null : _soundCue.OnPlayEnded;
            }
            set {
                if (_soundCue != null)
                {
                    _soundCue.OnPlayEnded = value;
                }
            }
        }
        public Action<SoundCue> OnPlayCueEnded
        {
            get {
                return _soundCue == null ? null : _soundCue.OnPlayCueEnded;
            }
            set {
                if (_soundCue != null)
                {
                    _soundCue.OnPlayCueEnded = value;
                }
            }
        }
        public SoundObject AudioObject
        {
            get {
                return _soundCue == null ? null : _soundCue.AudioObject;
            }
            set {
                if (_soundCue != null)
                {
                    _soundCue.AudioObject = value;
                }
            }
        }

        public SoundCueData Data { get { return _data; } }

        public bool IsPlaying
        {
            get {
                return _soundCue != null && _soundCue.IsPlaying;
            }
        }

        public int ID
        {
            get {
                return _soundCue == null ? -999 : _soundCue.ID;
            }
        }

        public void Play(SoundCueData data)
        {
            _data = data;
            if (_soundCue == null)
            {
                Debug.LogError("NO SOUND CUE to play!!!");
                return;
            }
            _soundCue.Play(data, this);
        }

        public void Pause()
        {
            if (_soundCue == null)
            {
                Debug.LogError("NO SOUND CUE to pause!!!");
                return;
            }
            _soundCue.Pause();
        }

        public void Resume()
        {
            if (_soundCue == null)
            {
                Debug.LogError("NO SOUND CUE to Resume!!!");
                return;
            }
            _soundCue.Resume();
        }

        public void Stop(bool shouldCallOnFinishedCue = true)
        {
            if (_soundCue == null)
            {
                Debug.LogError("NO SOUND CUE to Stop!!!");
                return;
            }
            _soundCue.Stop(shouldCallOnFinishedCue);
        }
        #endregion
    }
}
```

3.25 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Sound↵Object.cs File Reference

Classes

- class OSSC.SoundObject

Used by the SoundCue. Controls the AudioSource.

Namespaces

- namespace OSSC

3.26 SoundObject.cs

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

namespace OSSC
{
    [RequireComponent(typeof(AudioSource))]
    public class SoundObject : MonoBehaviour, IPoolable
    {

        public System.Action<SoundObject> OnFinishedPlaying;

        #region private fields
        private string _id;
        private AudioClip _clip;
        private AudioSource _source;

        private Coroutine _playingRoutine;
        private bool _isPaused;

        private bool _isFree = true;
        private PrefabBasedPool _pool;
        private float _fadeInTime;
        private float _fadeOutTime;
        private float _volume;
        private float _pitch;
        private bool _isDespawnOnFinishedPlaying = true;
        #endregion

        #region Public methods and properties
        public bool isDespawnOnFinishedPlaying {
            get { return _isDespawnOnFinishedPlaying; }
            set { _isDespawnOnFinishedPlaying = value; }
        }

        public string clipName
        {
            get
            {
                return _clip != null ? _clip.name : "NONE";
            }
        }

        public AudioSource source
        {
            get { return _source; }
        }

        public string ID {
            get { return _id; }
        }

        public void Setup(string id, AudioClip clip, float volume, float fadeInTime = 0f, float fadeOutTime = 0f, AudioMixerGroup mixer = null, float pitch = 1f)
        {
            _id = id;
            _clip = clip;
            gameObject.name = _id;
            if (_source == null)
                _source = GetComponent<AudioSource>();

            _source.volume = 0;
            _source.time = 0f;
            _source.outputAudioMixerGroup = mixer;
            _volume = volume;
            _pitch = pitch;
            _fadeInTime = fadeInTime;
            _fadeOutTime = fadeOutTime;
        }

        public void Play()
        {
            if (_source == null)
                _source = GetComponent<AudioSource>();
            _source.clip = _clip;
            gameObject.SetActive(true);
            _source.pitch = _pitch;
            StartCoroutine(FadeRoutine(_fadeInTime, _volume));
            _source.Play();
            _isFree = false;
            _playingRoutine = StartCoroutine(PlayingRoutine());
        }

        public void Pause()
        {
            if (_source == null)
                return;
            _isPaused = true;
            _source.Pause();
        }

        public void Resume()
        {
            if (_source == null)
                return;
            _source.Play();
            _isPaused = false;
        }

        public void Stop()
        {
            if (_playingRoutine == null)
                return;

            StartCoroutine(StopRoutine());
        }

        [ContextMenu("Test Play")]
        private void TestPlay()
        {
            Play();
        }
        #endregion

        private IEnumerator FadeRoutine(float fadeTime, float value)
        {
            if (fadeTime < 0.1f)
            {
                _source.volume = value;
                yield break;
            }

            float initVal = _source.volume;
            float fadeSpeed = 1f / (fadeTime / Time.deltaTime);
            for (float t = 0f; t < 1f; t += fadeSpeed)
            {
                float val = Mathf.SmoothStep(initVal, value, t);
                _source.volume = val;
                yield return null;
            }

            _source.volume = value;
        }

        private IEnumerator StopRoutine()
        {
            StopCoroutine(_playingRoutine);
            yield return StartCoroutine(FadeRoutine(_fadeOutTime, 0f));
            _source.Stop();
            _source.clip = null;
            _playingRoutine = null;
            _isFree = true;
            _volume = 0f;
            _source.time = 0f;
            _source.pitch = 1f;

            if (isDespawnOnFinishedPlaying)
                _pool.Despawn(gameObject);

            if (OnFinishedPlaying != null)
            {
                OnFinishedPlaying(this);
            }
        }

        private IEnumerator PlayingRoutine()
        {
            while (true)
            {
                yield return null;
                float fadeOutTrigger = _source.clip.length - _fadeOutTime;
                if (_source.time >= fadeOutTrigger)
                {
                    yield return StartCoroutine(FadeRoutine(_fadeOutTime, 0f));
                }
                if (!_source.isPlaying && !_isPaused)
                {
                    break;
                }
            }

            _source.clip = null;
            _playingRoutine = null;
            _isFree = true;
            _volume = 0f;
            _source.time = 0f;

            if (isDespawnOnFinishedPlaying)
                _pool.Despawn(gameObject);

            if (OnFinishedPlaying != null)
            {
                OnFinishedPlaying(this);
            }
        }

        #region IPoolable methods
        PrefabBasedPool IPoolable.pool {
            get { return _pool; }
            set { _pool = value; }
        }

        public bool IsFree()
        {
            return _isFree;
        }
        #endregion
    }

}
```

3.27 F:/git-projects/audio-controller-unity/AudioController/Assets/OSSC/Source/Sound↩ Tags.cs File Reference

Classes

- class OSSC.SoundTags

  Used By the SoundController for tagging SoundItems.
- struct OSSC.TagData

  Used by the SoundTags to save Tags.

Namespaces

- namespace OSSC

3.28 SoundTags.cs

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

namespace OSSC
{
    [System.Serializable]
    public class SoundTags
    {
        #region Private fields
        [SerializeField]
        private List<TagData> _tagsData;

        private int _lastID = 0;
        #endregion

        public SoundTags()
        {
            _tagsData = new List<TagData>();
        }

        #region Public methods and properties

        public TagData[] ToArray()
        {
            return _tagsData.ToArray();
        }

        public string[] ToArrayNames()
        {
            string[] names = new string[_tagsData.Count];
            for (int i = 0; i < names.Length; i++)
            {
                names[i] = _tagsData[i].name;
            }

            return names;
        }

        public int[] ToArrayIDs()
        {
            int[] ids = new int[_tagsData.Count];
            for (int i = 0; i < ids.Length; i++)
            {
                ids[i] = _tagsData[i].ID;
            }

            return ids;
        }

        public TagData GetTagDataByName(string name)
        {
            return _tagsData.Find(data => data.name.Equals(name.ToLower()));
        }

        public TagData GetTagDataByID(int ID)
        {
            return _tagsData.Find(data => data.ID.Equals(ID));
        }

        public int GetTagIDByName(string name)
        {
            TagData result = _tagsData.Find(data => data.name.Equals(name.ToLower()));
            if (string.IsNullOrEmpty(result.name))
                return -1;

            return result.ID;
        }

        public string GetTagNameByID(int ID)
        {
            TagData result = _tagsData.Find(data => data.ID.Equals(ID));
            if (string.IsNullOrEmpty(result.name))
                return string.Empty;

            return result.name;
        }

        public void SetTag(string name)
        {
            if (string.IsNullOrEmpty(name))
                return;

            string nameLowercase = name.ToLower();
            TagData result = _tagsData.Find(data => data.name.Equals(nameLowercase));
            if (string.IsNullOrEmpty(result.name) == false)
                return;

            result.name = nameLowercase;
            result.ID = _lastID;
            _lastID += 1;
            _tagsData.Add(result);
        }

        public void RemoveByTag(TagData data)
        {
            _tagsData.Remove(data);
        }
        #endregion
    }

    [System.Serializable]
    public struct TagData
    {
        public string name;
        public int ID;
    }
}
```

Index