Table of Contents

Introduction

Overview

Stem is a lightweight audio manager. It introduces the concept of sound and music bank as persistent storage for organized audio clips and requires no scene setup.

Features

- Optimized for runtime use: great performance, zero memory allocations per frame.
- Persists during runtime: you don't have to think about managing game objects.
- Requires no scene setup: just create a few assets, fill it with data and that's it.
- Works with multiple scenes: no prefabs, no additional game objects.
- Integrates with Unity 5 Audio: easily connect sound buses and music players to mixer groups.
- Saves time with the batch import feature.
- Supports sound variations: use multiple audio clips, randomize volume/pitch/delay for each clip.
- Supports 3d sound: tune the most important audio source settings within the bank.
- Supports layered music playback with independent playback controls.

Package Contents

The package consists of three folders:

1. *Documentation* — contains pdf file with manual and API reference. It fully matches the content of this site.
2. *Samples* — simple scenes that covers all Stem features.
3. *Scripts* — source code of the package.

Stem source code is divided into four parts:

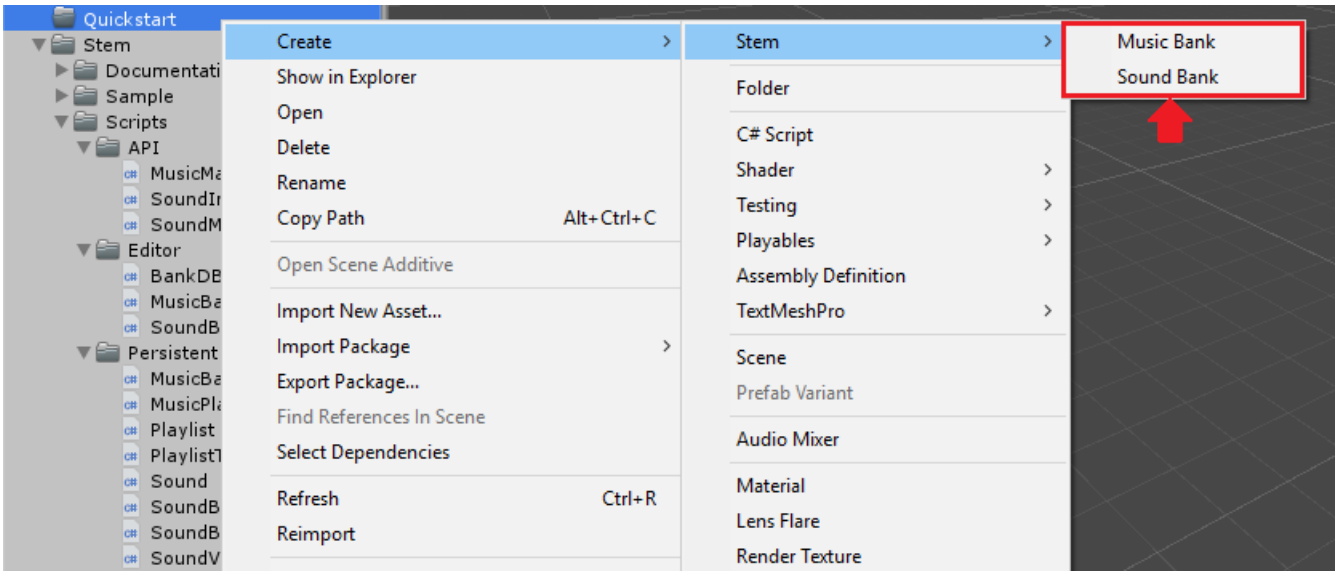
1. *API* — public API classes. Use them to perform operations in Stem.
2. *Persistent* — public data classes. Will be stored inside sound and music bank assets.
3. *Editor* — internal editor classes (inspectors for sound and music banks, scene postprocessor). Will not be included during the build stage.
4. *Runtime* — internal logic classes. Will be automatically created on demand. Don't use them manually.

Quickstart

Create sound and music bank assets

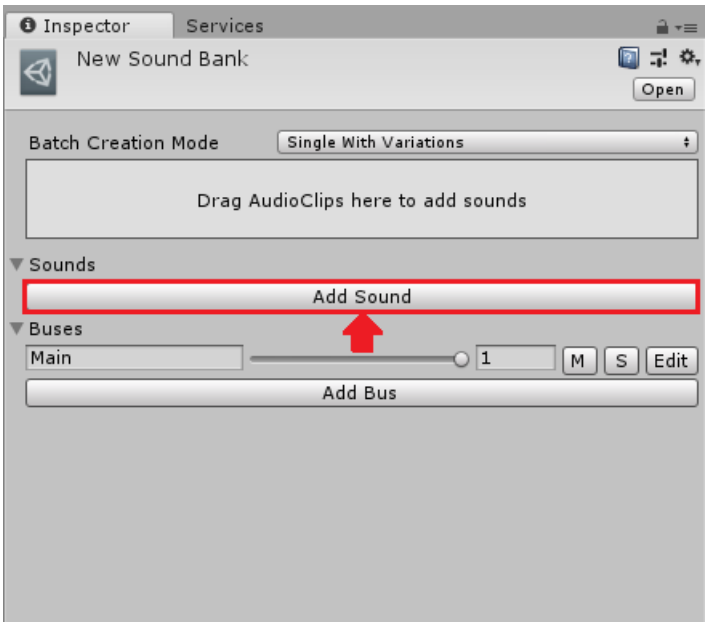
Press right mouse button on some folder within Assets folder.

1. Choose "Create -> Stem -> Music Bank" to create a music bank.
2. Choose "Create -> Stem -> Sound Bank" to create a sound bank.

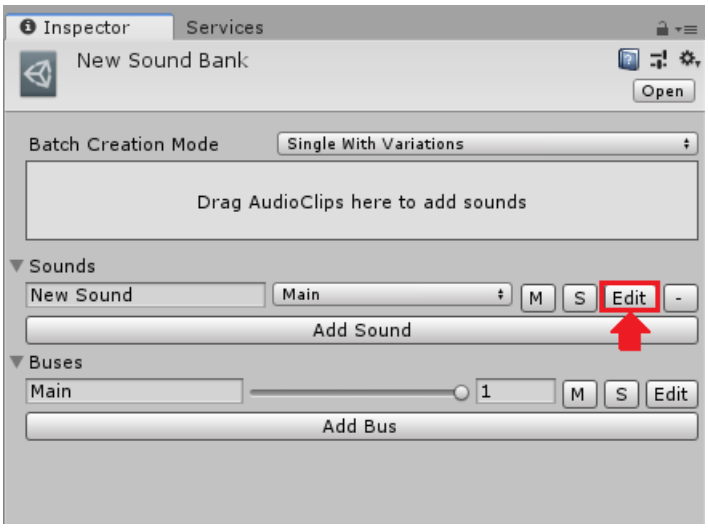


Setup sound bank

Select a sound bank asset and add a new sound by pressing the "Add Sound" button in the inspector window.

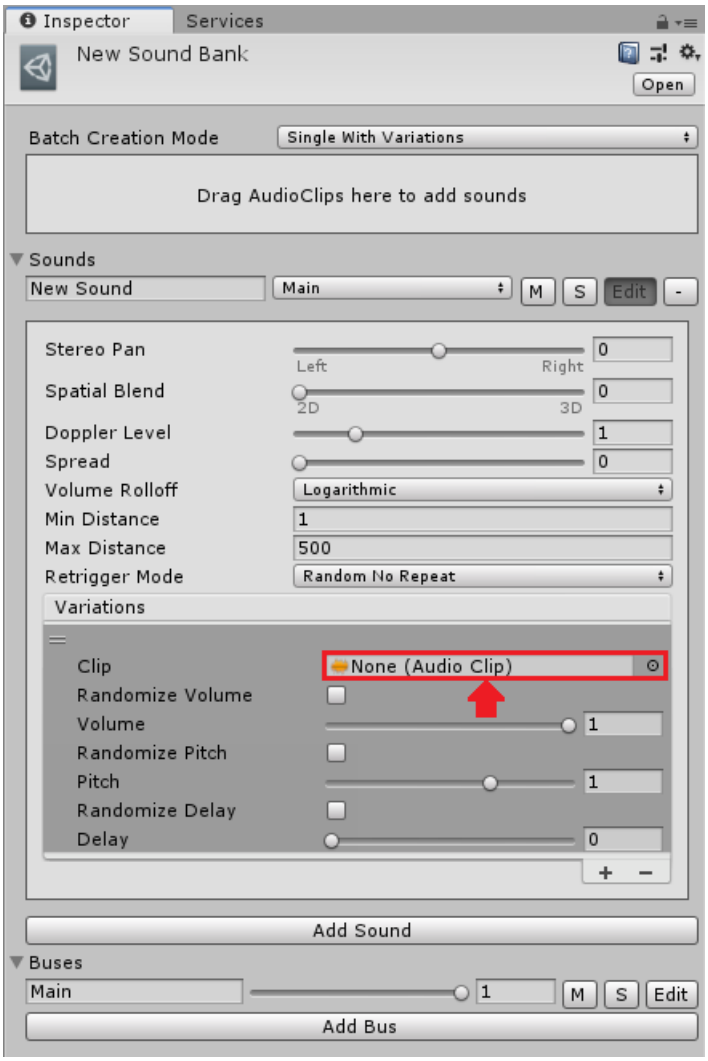


Expand sound settings by toggling the "Edit" button.

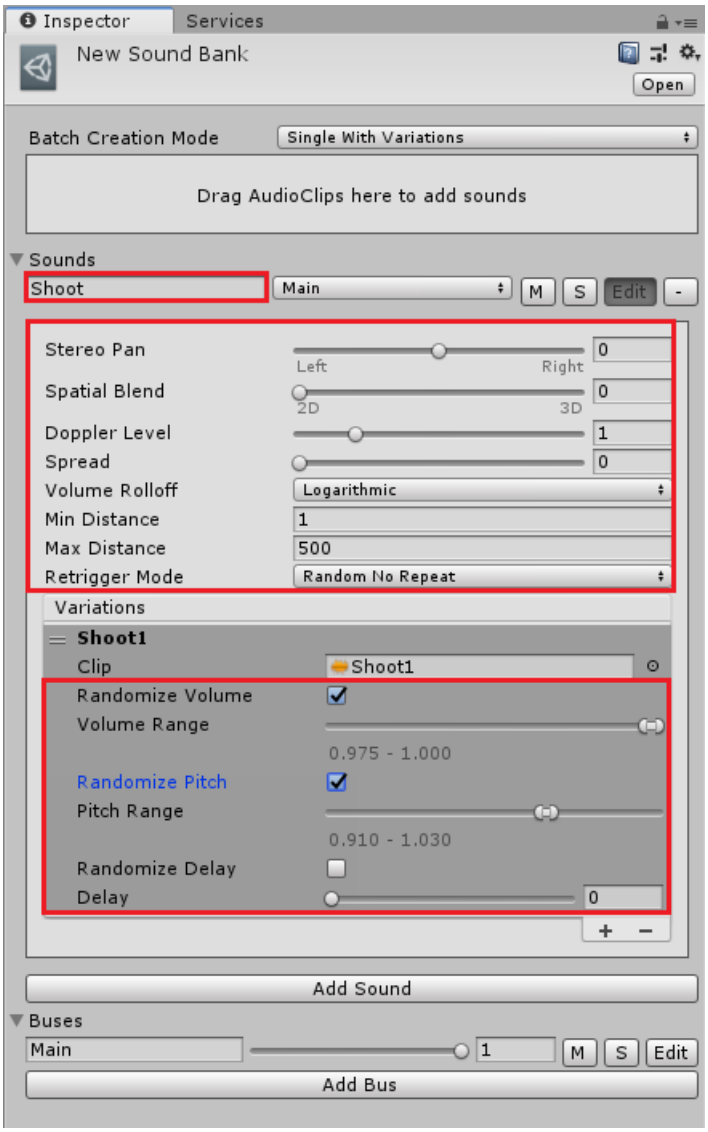


Add the first variation to the sound by pressing the "+" button in the "Variations" list.

Assign an audio clip to the variation by dragging it into the "Clip" property.

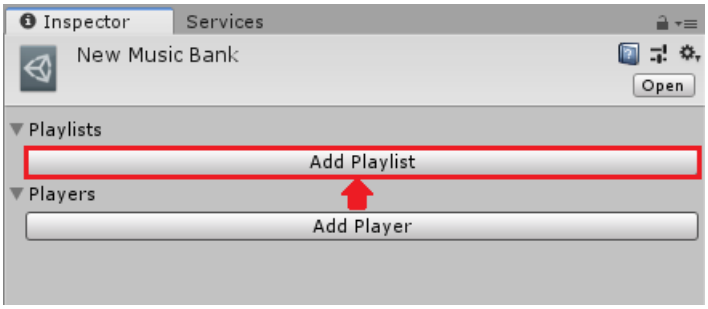


Set sound name, adjust parameters and randomization settings.



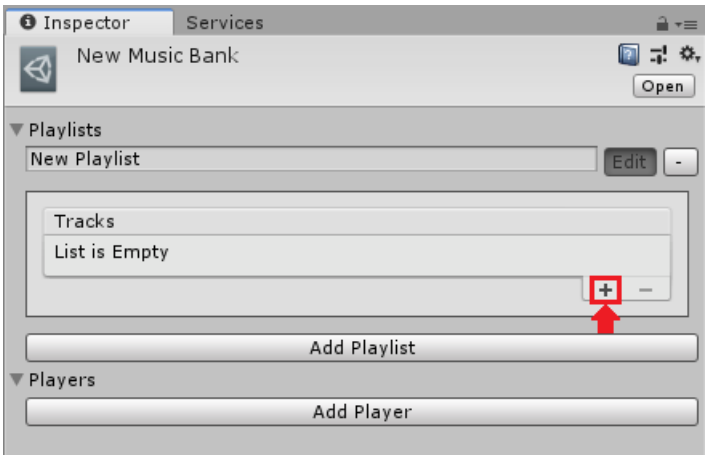
Setup music bank

Select a music bank asset and add a new playlist by pressing the "Add Playlist" button in the inspector window.

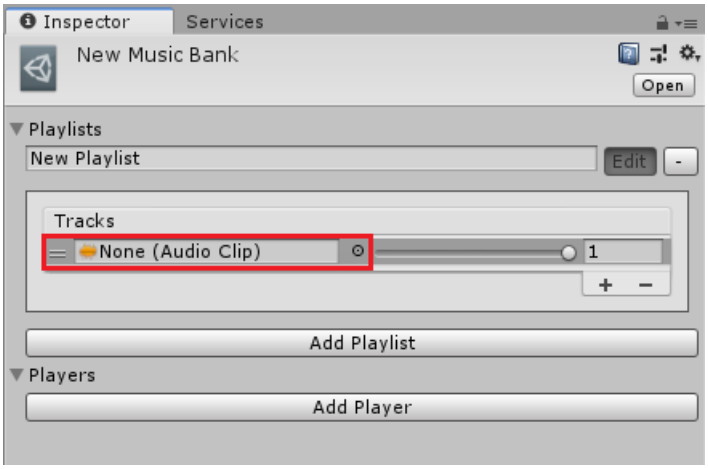


Expand playlist settings by toggling the "Edit" button.

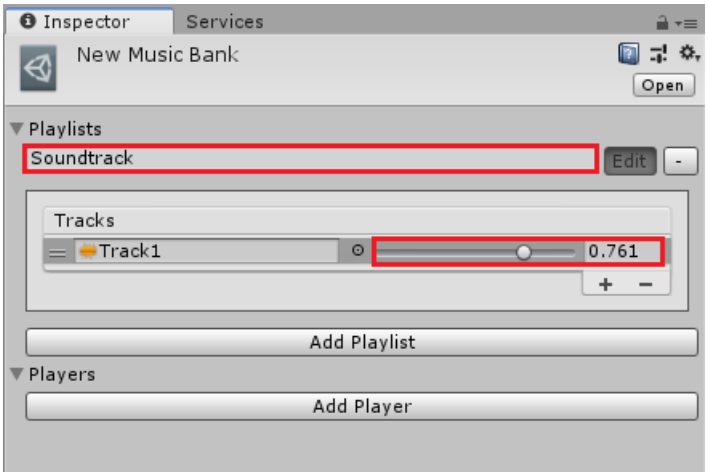
Add the first track to the playlist by pressing the "+" button in the "Tracks" list.



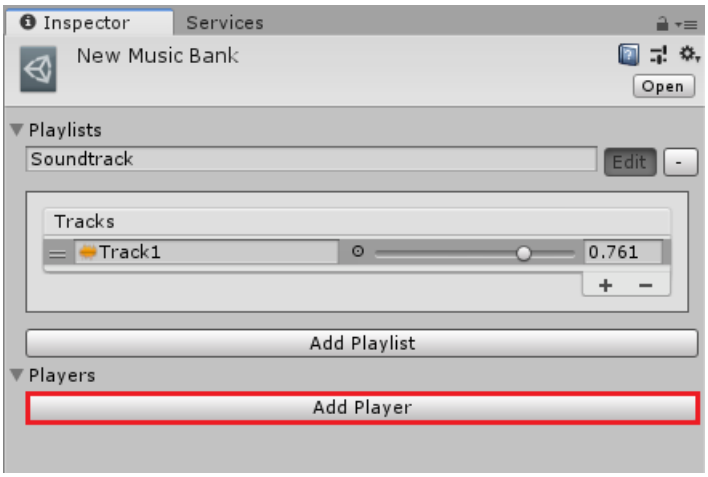
Assign an audio clip to the track.



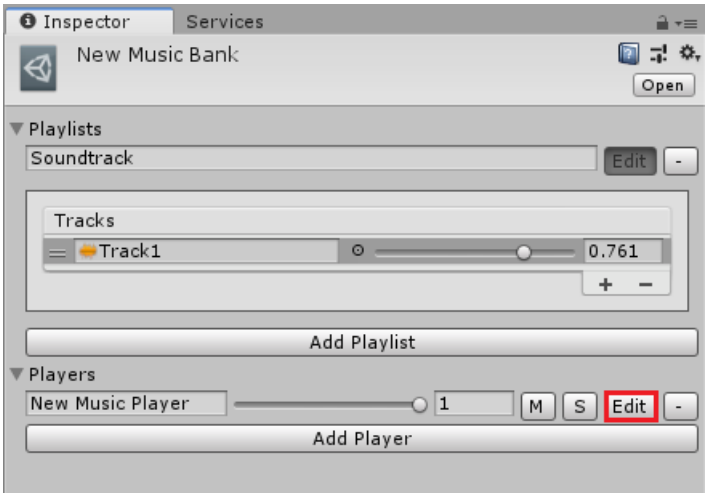
Set playlist name, adjust track volume.



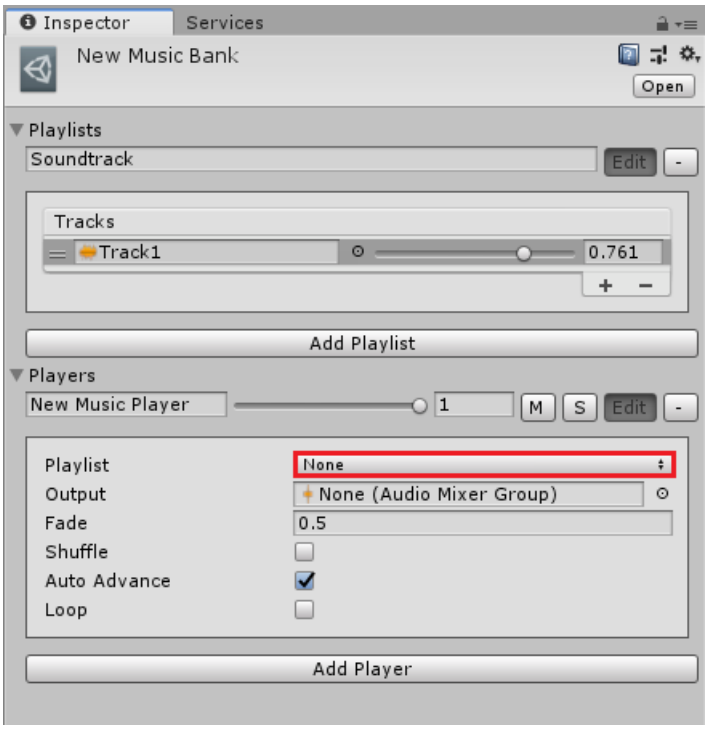
Add a new music player by pressing the "Add Player" button in the inspector window.



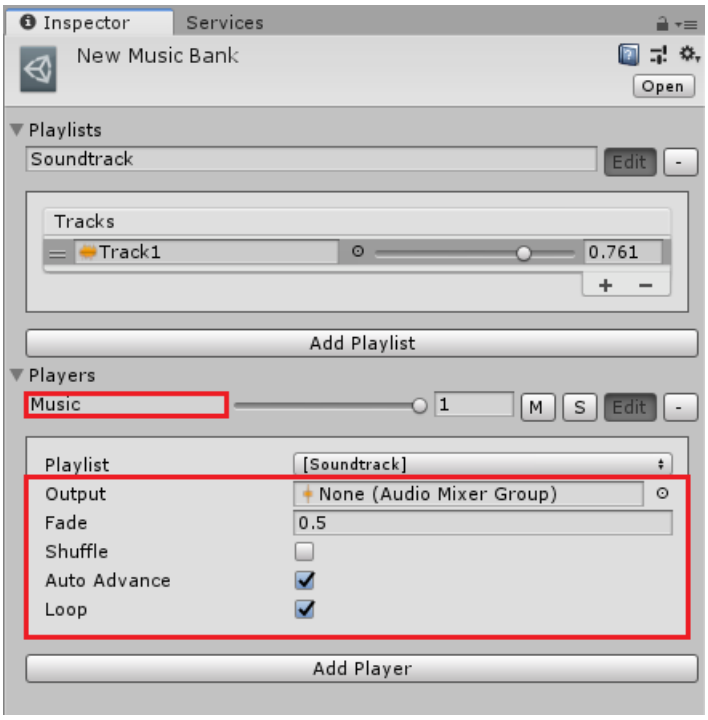
Expand music player settings by toggling the "Edit" button.



Choose the playlist from the "Playlist" drop-down menu.



Set music player name, adjust parameters.



Use in the code

Play one-shot sounds by calling Stem.SoundManager.Play.

```
using UnityEngine;

public class SimpleGun : MonoBehaviour
{
    public float timeBetweenBullets = 0.15f;
    private float timer = 0.0f;

    private void Update()
    {
        timer += Time.deltaTime;

        if(Input.GetButton("Fire1") && timer >= timeBetweenBullets)
        {
            timer = 0.0f;
            Stem.SoundManager.Play("Shoot");
        }
    }
}
```

Begin music playback during level startup by calling Stem.MusicManager.Play.

```
using UnityEngine;

public class LevelStartup : MonoBehaviour
{
    private void Start()
    {
        Stem.MusicManager.Play("Music");
    }
}
```

Music Bank

API Reference

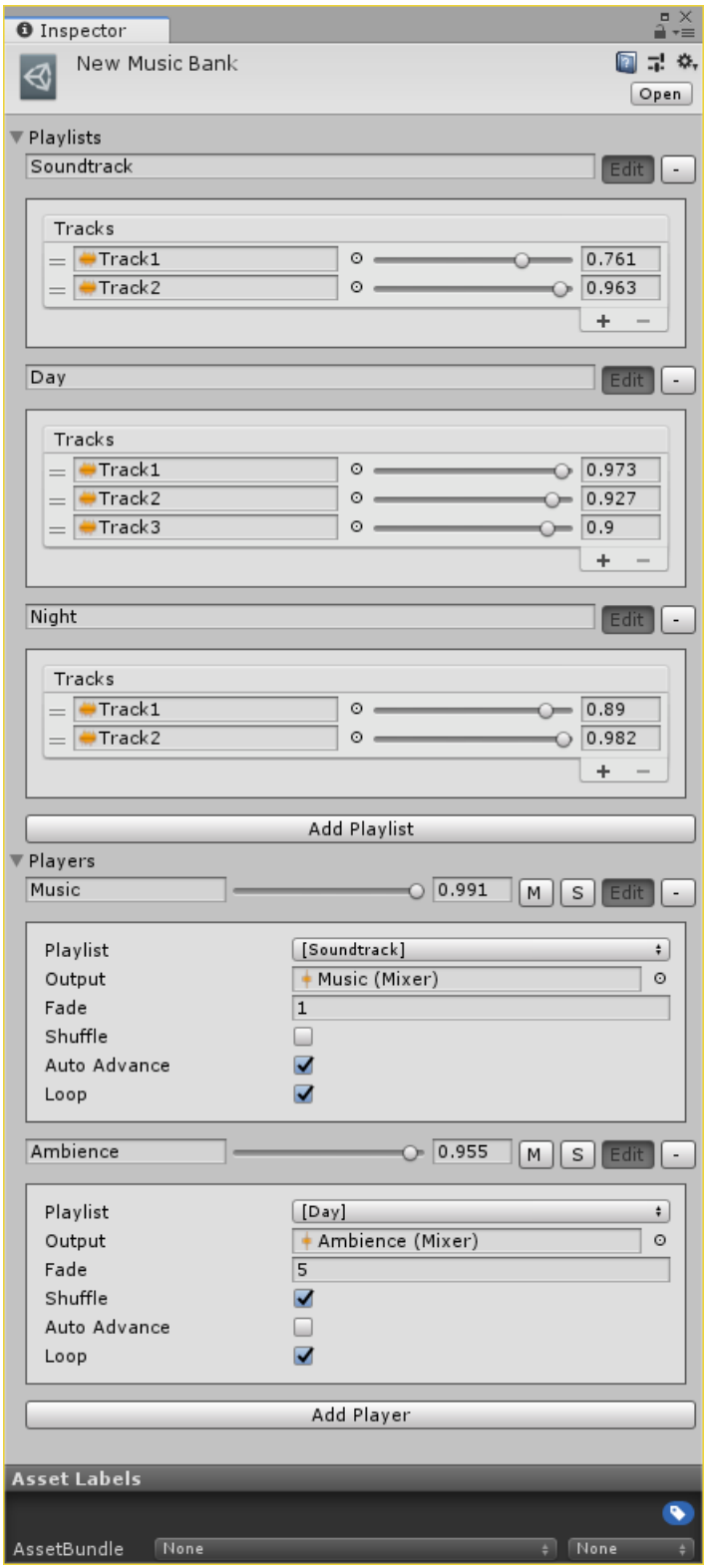
Description

Music bank is a collection of playlists and music players. It's stored inside the project *Assets* folder. Once filled with the content, it's ready to use in the code.

It's possible to have several music banks. Stem will automatically search for the corresponding music bank during the playlist or music player lookup. In the case of name collisions, if multiple banks have music players or playlists with the same name, primary music bank (which you can only set in the code) will be checked first. Within a bank, the first occurrence of music player or playlist will be used.

All music tracks must be organized into playlists. It could be thematic playlists, i.e. gameplay music, menu music, outdoor ambience, etc. or big single playlist containing all the tracks.

Each music player can only play a single playlist. Music bank allows doing a layered music playback by creating multiple music players and filling them with different playlists.



Properties

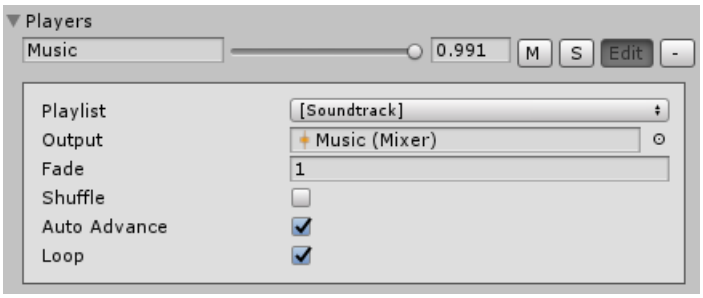
PROPERTY	DESCRIPTION
Playlists	A collection of playlists.
Players	A collection of music players.

Music Player

API Reference

Description

A music player is a part of the music bank. It represents an entry point for music playback and defines how it will play the playlist tracks. It also integrates into a Unity Audio Mixer.



Properties

PROPERTY	DESCRIPTION
Name	The name of the music player.
Volume	The master volume of the music player.
Playlist	The drop-down menu allowing to select a playlist. *None* option will require to set the playlist in order to play.
Output	The reference to AudioMixerGroup. Please refer to Unity Manual for details.
Fade	The crossfade duration used by the music player during transitions between tracks or playback state changes.
Shuffle	The flag indicating whether the music player should play tracks in random order.
Auto Advance	The flag indicating whether the music player should auto advance to next playlist track it finishes.
Loop	The flag indicating whether the music player should repeat playlist tracks after they finish.

Loop and *Auto Advance* flags work in pair. There are four possible combinations:

LOOP	AUTO ADVANCE	BEHAVIOUR
False	False	Play the current track once and then stop.
False	True	Play all playlist tracks once and then stop.
True	False	Play the current track in a loop and never stop.
True	True	Play all playlist tracks in a loop and never stop.

Playlist

API Reference

Description

A playlist is a part of the music bank. It represents a collection of music tracks.

Properties

PROPERTY	DESCRIPTION
Name	The name of the playlist.

Track Properties

API Reference

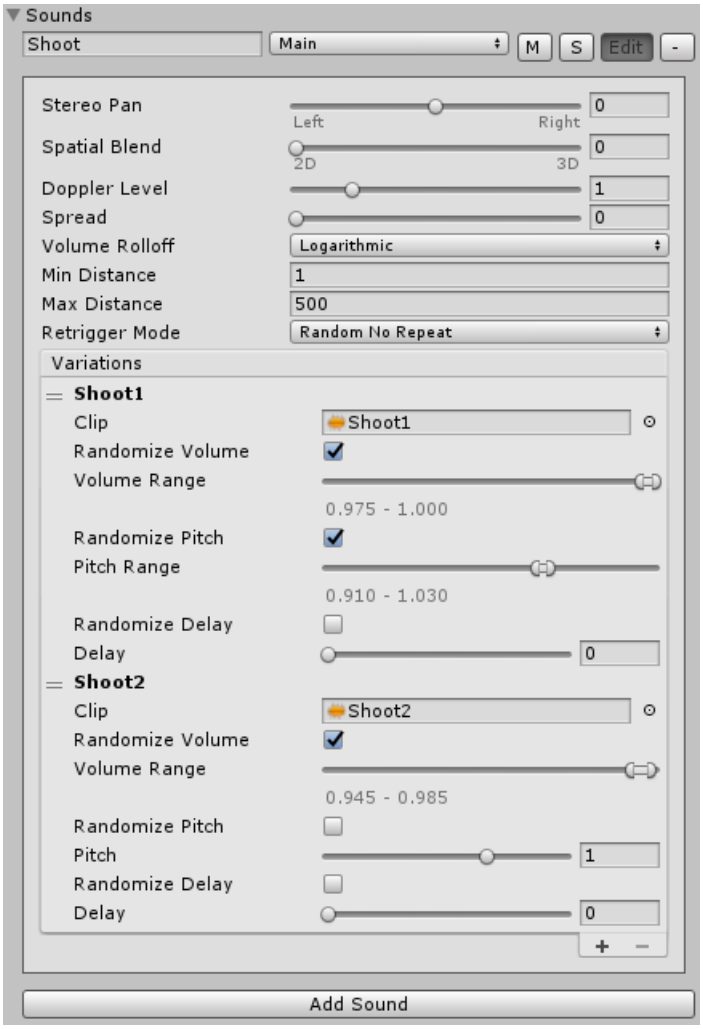
PROPERTY	DESCRIPTION
Clip	The reference to *Audio Clip*. Please refer to Unity Manual for details.
Volume	The master volume of the track.

Sound

API Reference

Description

A sound is a part of the sound bank. It represents a set of parameters defining how the sound will be mixed (most of them are duplicates from *Audio Source*). It could also have multiple audio clips serving as variations. Each sound variation can be randomized in volume, pitch and delay.



Properties

PROPERTY	DESCRIPTION
Name	The name of the sound.
Stereo Pan	The stereo panning parameter defining sound position in a stereo way (left or right).
Spatial Blend	The spatial blend parameter defining how much the sound is affected by 3d spatialisation calculations (attenuation, doppler etc).
Doppler Level	The scale of doppler effect that will be applied to the sound (if is set to 0, then no effect is applied).
Spread	The spread angle (in degrees) of a 3d stereo or multichannel sound in speaker space.

PROPERTY	DESCRIPTION
Volume Rolloff	The attenuation mode defining how sound volume will be lowered over the distance.
Min Distance	The parameter defining the boundary within which the sound won't get any louder. Outside *Min Distance* it will begin to attenuate.
Max Distance	The parameter defining the boundary outside which the sound will be inaudible or stop attenuating depending on ** Volume Rolloff** value.
Retrigger Mode	The rule defining how the sound will play variations.

> *Stereo Pan*, *Spatial Blend*, *Doppler Level*, *Spread*, *Min Distance*, *Max Distance* parameters duplicate corresponding parameters from *Audio Source*. Please refer to Unity Manual for details.

Variation Properties

API Reference

PROPERTY	DESCRIPTION
Clip	The reference to *AudioClip*. Please refer to Unity Manual for details.
Randomize Volume	The flag indicating whether the sound variation should be randomized in volume.
Volume / Volume Range	The volume of the sound variation. Depending on *Randomize Volume* flag it's either a set value or a random value from the range.
Randomize Pitch	The flag indicating whether the sound variation should be randomized in pitch.
Pitch / Pitch Range	Amount of change in pitch due to slowdown/speed up of the *AudioClip*. Depending on *Randomize Pitch* flag it's either a set value or a random value from the range.
Randomize Delay	The flag indicating whether the sound variation should be randomized in delay.
Delay / Delay Range	The playback delay in seconds. Depending on *Randomize Delay* flag it's either a set value or a random value from the range.

Sound Bank

API Reference

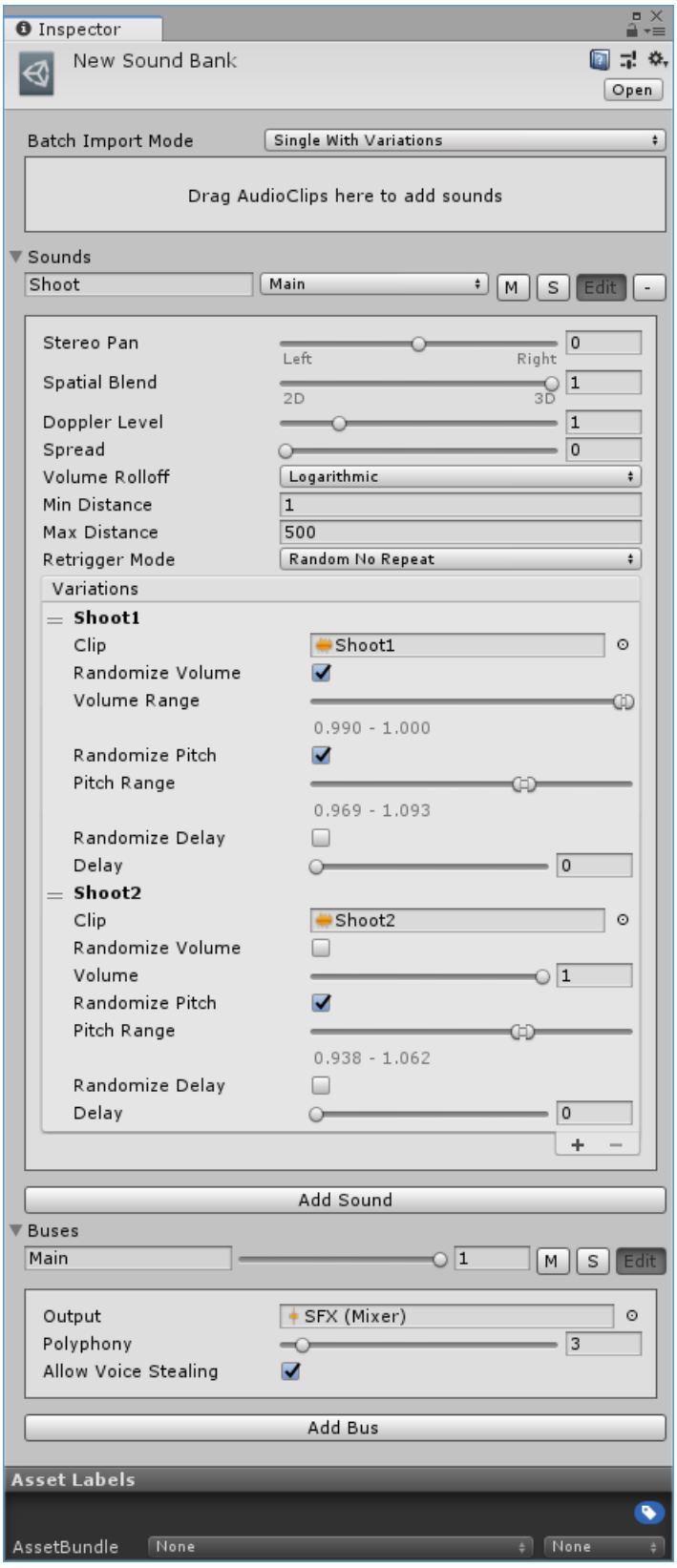
Description

Sound bank is a collection of sounds and sound buses. It's stored inside the project Assets folder. Once filled with the content, it's ready to use in the code.

It's possible to have several sound banks. Stem will automatically search for the corresponding sound bank during the sound or sound bus lookup. In the case of name collisions, if multiple banks have sounds or sound buses with the same name, primary sound bank (which you can only set in the code) will be checked first. Within a bank, the first occurrence of sound or sound bus will be used.

All sound effects must be organized into sounds with multiple variations. Each sound must have a sound bus assigned. It could be thematic sound buses, i.e. character, enemies, gunshots, etc. or just a single Master sound bus.

Each sound bus controls how many sounds can be played simultaneously.



Properties

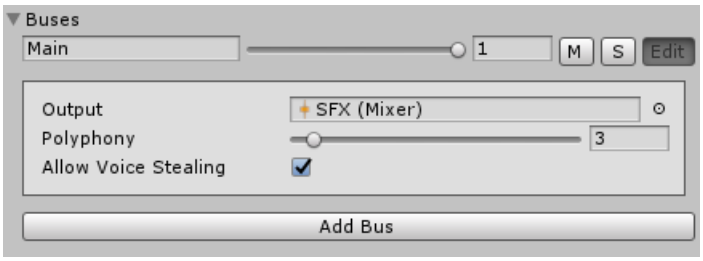
PROPERTY	DESCRIPTION
Sounds	A collection of sounds.
Buses	A collection of sound buses.

Sound Bus

API Reference

Description

A sound bus is a part of the sound bank. It groups sound and define the limit of simultaneously playing sounds. It also integrates into a Unity Audio Mixer.



Properties

PROPERTY	DESCRIPTION
Name	The name of the sound bus.
Output	The reference to *Audio Mixer Group*. Please refer to Unity Manual for details.
Polyphony	The number of maximum allowed simultaneously playing sounds in the sound bus.
Allow Voice Stealing	The flag indicating whether the sound bus can stop the oldest playing sound and play the new one if *Polyphony* limit is exceeded.

Music Manager

API Reference

Description

Music Manager is the entry point for music playback and music bank management. It does not require to create any additional game objects in order to use it. Music playback and crossfade logic is handled by the music manager runtime.

Track controls

Call Stem.MusicManager.SetPlaylist to assign a playlist to a music player.

```
// Note that crossfade duration value from the music player will be used by default.
string musicPlayerName = "Music";
string playlistName = "Soundtrack";

// Will do a lookup for music player with the name "Music" in all music banks
// then do the same lookup for playlist with the name "Soundtrack".
Stem.MusicManager.SetPlaylist(musicPlayerName, playlistName);

// It's possible to override crossfade duration.
float newCrossfade = 5.0f;
Stem.MusicManager.SetPlaylist(musicPlayerName, playlistName, newCrossfade);
```

Call these methods to set a desired track to a music player:

- Stem.MusicManager.Next
- Stem.MusicManager.Prev
- Stem.MusicManager.Seek.

```
// Note that crossfade duration value from the music player will be used by default.
string musicPlayerName = "Music";
string trackName = "Level1 Theme";

// Will advance music player to next track
Stem.MusicManager.Next(musicPlayerName);

// Will advance music player to previous track
Stem.MusicManager.Prev(musicPlayerName);

// Will advance music player to a target track
Stem.MusicManager.Seek(musicPlayerName, trackName);

// It's possible to override crossfade duration.
float newCrossfade = 5.0f;
Stem.MusicManager.Next(musicPlayerName, newCrossfade);
Stem.MusicManager.Prev(musicPlayerName, newCrossfade);
Stem.MusicManager.Seek(musicPlayerName, trackName, newCrossfade);
```

Music playback

Call these methods to change playback state of a music player:

- Stem.MusicManager.Play
- Stem.MusicManager.Stop
- Stem.MusicManager.Pause
- Stem.MusicManager.UnPause

```
// Note that crossfade duration value from the music player will be used by default.
string musicPlayerName = "Music";

// Will start playing current track of the music player.
Stem.MusicManager.Play(musicPlayerName);

// Will pause playing current track of the music player.
Stem.MusicManager.Pause(musicPlayerName);

// Will resume playing current track of the music player.
Stem.MusicManager.UnPause(musicPlayerName);

// Will stop playing current track of the music player.
Stem.MusicManager.Stop(musicPlayerName);

// It's possible to override crossfade duration.
float newCrossfade = 5.0f;
Stem.MusicManager.Play(musicPlayerName, newCrossfade);
Stem.MusicManager.Pause(musicPlayerName, newCrossfade);
Stem.MusicManager.UnPause(musicPlayerName, newCrossfade);
Stem.MusicManager.Stop(musicPlayerName, newCrossfade);
```

Sound Instance

API Reference

Description

Sound Instance is a single playing audio source. Most sound instances are managed by the Sound Manager and are not used directly.

However, this class is very useful for custom mixing logic. There are four main use cases:

1. Playing looped sounds
2. Using multiple sounds within a sound instance
3. Changing volume and pitch during playback
4. Attaching to another game object

Playing looped sounds

Use the Stem.SoundInstance.Looped flag to play looped sound.

```
using UnityEngine;

public class TVNoise : MonoBehaviour
{
    private Stem.SoundInstance soundInstance = null;

    private void OnEnable()
    {
        if (soundInstance != null)
            return;

        // Will do a lookup for sound with the name "TVNoise" in all sound banks
        // and return a reference to a sound instance from the sound pool.
        soundInstance = Stem.SoundManager.GrabSound("TVNoise");

        // Play looped
        soundInstance.Looped = true;
        soundInstance.Play();
    }

    private void OnDisable()
    {
        if (soundInstance == null)
            return;

        soundInstance.Stop();

        // Return the instance to the sound pool
        Stem.SoundManager.ReleaseSound(soundInstance);
        soundInstance = null;
    }
}
```

Using multiple sounds within a sound instance

Use the Stem.SoundInstance.Sound property to play different sounds.

```
using UnityEngine;

public class Draggable : MonoBehaviour
{
    private Stem.SoundInstance soundInstance = null;
    private Stem.Sound dragStart = null;
    private Stem.Sound dragEnd = null;
    private Stem.Sound dragLoop = null;
    private bool dragging = false;

    private void Awake()
    {
        // Grab an empty sound instance from the sound pool.
        soundInstance = Stem.SoundManager.GrabSound();

        dragStart = Stem.SoundManager.GetSound("DragStart");
        dragEnd = Stem.SoundManager.GetSound("DragEnd");
        dragLoop = Stem.SoundManager.GetSound("DragLoop");
    }

    private void OnDestroy()
    {
        soundInstance.Stop();

        // Return the instance to the sound pool
        Stem.SoundManager.ReleaseSound(soundInstance);
        soundInstance = null;
    }

    private void Update()
    {
        if (dragging && soundInstance.Sound == dragStart && !soundInstance.Playing)
        {
            // Play looped
            soundInstance.Sound = dragLoop;
            soundInstance.Looped = true;
            soundInstance.Play();
        }
    }

    public void StartDrag()
    {
        // Play one-shot
        soundInstance.Sound = dragStart;
        soundInstance.Looped = false;
        soundInstance.Play();

        dragging = true;
    }

    public void EndDrag()
    {
        // Play one-shot
        soundInstance.Sound = dragEnd;
        soundInstance.Looped = false;
        soundInstance.Play();

        dragging = false;
    }
}
```

Changing volume and pitch during playback

Use Stem.SoundInstance.Volume and Stem.SoundInstance.Pitch properties to change parameters during playback.

```
using UnityEngine;

public class Siren : MonoBehaviour
{
    public float frequency = 440.0f;

    public float baseVolume = 0.8f;
    public float additionalVolume = 0.2f;

    public float basePitch = 1.0f;
    public float additionalPitch = 0.2f;

    private Stem.SoundInstance soundInstance = null;

    private void Awake()
    {
        // Will do a lookup for sound with the name "Siren" in all sound banks
        // and return a reference to a sound instance from the sound pool.
        soundInstance = Stem.SoundManager.GrabSound("Siren");

        // Play looped
        soundInstance.Looped = true;
        soundInstance.Play();
    }

    private void OnDestroy()
    {
        soundInstance.Stop();

        // Return the instance to the sound pool
        Stem.SoundManager.ReleaseSound(soundInstance);
        soundInstance = null;
    }

    private void Update()
    {
        // Calculate current sine wave sample
        float timeNow = Time.realtimeSinceStartup;
        float sample = Mathf.Sin(timeNow * frequency);

        // Modulate pitch and volume
        soundInstance.Volume = baseVolume + additionalVolume * sample;
        soundInstance.Pitch = basePitch + additionalPitch * sample;
    }
}
```

Attaching to another game object

Use the Stem.SoundInstance.Transform property to attach sound instance to another game object.

```
using UnityEngine;

public class Rocket : MonoBehaviour
{
    private Stem.SoundInstance soundInstance = null;

    private void Awake()
    {
        // Will do a lookup for sound with the name "Engine" in all sound banks
        // and return a reference to a sound instance from the sound pool.
        soundInstance = Stem.SoundManager.GrabSound("Engine");

        // Play looped
        soundInstance.Looped = true;
        soundInstance.Play();

        // Attach to the game object
        soundInstance.Transform.parent = transform;
    }

    private void OnDestroy()
    {
        soundInstance.Stop();

        // Return the instance to the sound pool
        Stem.SoundManager.ReleaseSound(soundInstance);
        soundInstance = null;
    }
}
```

Sound Manager

API Reference

Description

Sound Manager is the entry point for sound playback and sound bank management. It does not require to create any additional game objects in order to use it.

All currently playing sounds represented by the Sound Instance class. There are two use cases:

1. One-shot sounds — they are played and managed internally by the sound manager. When playing a sound, Sound Manager will look up for available sound instance and use it for playback.
2. Custom user logic — the sound manager allows getting a reference to a sound instance from the sound pool for manual playback.

Managed one-shot sound instances

Call Stem.SoundManager.Play to play a one-shot sound.

```
// Note that volume, pitch and delay values from the sound will be used by default.

// Will do a lookup for sound with the name "Footstep" in all sound banks,
// create an internal sound instance and play it once.
Stem.SoundManager.Play("Footstep");

// It's possible to override sound instance volume, pitch and delay values.
float newVolume = 0.5f;
float newPitch = 0.9f;
float newDelay = 0.1f;
Stem.SoundManager.Play("Footstep", newVolume, newPitch, newDelay);
```

Call Stem.SoundManager.Play3D to play a one-shot sound in 3D space.

```
// Note that volume, pitch and delay values from the sound will be used by default.

// Will do a lookup for sound with the name "Explosion" in all sound banks,
// create an internal sound instance and play it once.
Vector3 hitPosition = new Vector3(100.0f, 0.0f, 0.0f);
Stem.SoundManager.Play3D("Explosion", hitPosition);

// Each method allow to override volume, pitch and delay values.
float newVolume = 0.5f;
float newPitch = 0.9f;
float newDelay = 0.1f;
Stem.SoundManager.Play3D("Explosion", hitPosition, newVolume, newPitch, newDelay);
```

Manual sound instances

Call Stem.SoundManager.GrabSound to get a reference to a sound instance from the sound pool. Once the sound instance is not needed anymore, call Stem.SoundManager.ReleaseSound to return it back to the sound pool.

```
// Note that volume, pitch and delay values from the sound will be used by default.

// Will do a lookup for sound with the name "GunShot" in all sound banks
// and return a reference to a sound instance from the sound pool.
Stem.SoundInstance soundInstance = Stem.SoundManager.GrabSound("GunShot");

// Play one-shot sound
soundInstance.Play();

// It's possible to override sound instance volume, pitch and delay values.
float newVolume = 0.5f;
float newPitch = 0.9f;
float newDelay = 0.1f;
soundInstance.Play(newVolume, newPitch, newDelay);

// Or play one-shot sound in 3D space
Vector3 hitPosition = new Vector3(100.0f, 0.0f, 0.0f);
soundInstance.Play3D(hitPosition);

// Also with overrides
soundInstance.Play3D(hitPosition, newVolume, newPitch, newDelay);

// Return the instance to the sound pool
Stem.SoundManager.ReleaseSound(soundInstance);
```

Global sound playback control

Call Stem.SoundManager.Pause, Stem.SoundManager.UnPause and Stem.SoundManager.Stop to pause, resume and stop all sound instances immediately.

```
// Get a reference to a sound instance from the sound pool
Stem.SoundInstance soundInstance = Stem.SoundManager.GrabSound("GunShot");

// Play one-shot sound manually
soundInstance.Play();

// Play one-shot sound automatically
Stem.SoundManager.Play("GunShot");

// Stop all playing sound instances (including ones from the sound pool)
Stem.SoundManager.Pause();

// Resume all playing sound instances (including ones from the sound pool)
Stem.SoundManager.UnPause();

// Stop all playing sound instances (including ones from the sound pool)
Stem.SoundManager.Stop();

// Return the instance to the sound pool
Stem.SoundManager.ReleaseSound(soundInstance);
```

Namespace Stem

Classes

MusicBank

The persistent storage for playlists and music players.

MusicManager

The main class for music playback and bank management.

MusicPlayer

The persistent storage for playback rules of a particular playlist.

Playlist

The persistent collection of playlist tracks.

PlaylistTrack

The persistent storage for music audio data.

Sound

The persistent storage for sound variations and the most important audio source settings.

SoundBank

The persistent storage for sounds and sound buses.

SoundBus

The persistent storage for playback rules of a group of sounds.

SoundInstance

The game object with audio source component. Used for manual playback and custom mixing logic.

SoundManager

The main class for sound playback and bank management.

SoundVariation

The persistent storage for sound effect audio data.

Enums

AttenuationMode

Defines how sound volume will be lowered over the distance.

RetriggerMode

Defines how sound will play its variations.

SoundBatchImportMode

Defines how new sounds will be created after the drag-drop event.

Enum AttenuationMode

Defines how sound volume will be lowered over the distance.

Namespace: Stem

Assembly: Stem.dll

Syntax

```
[Serializable]
public enum AttenuationMode
```

Fields

NAME	DESCRIPTION
Linear	A linear rolloff.
Logarithmic	A real-world rolloff.

Class MusicBank

The persistent storage for playlists and music players.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.ScriptableObject
MusicBank

Implements

UnityEngine.ISerializationCallbackReceiver

Inherited Members

UnityEngine.ScriptableObject.SetDirty()
UnityEngine.ScriptableObject.CreateInstance(System.String)
UnityEngine.ScriptableObject.CreateInstance(System.Type)
UnityEngine.ScriptableObject.CreateInstance<T>()
UnityEngine.Object.GetInstanceID()
UnityEngine.Object.GetHashCode()
UnityEngine.Object.Equals(System.Object)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Vector3, UnityEngine.Quaternion)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Vector3, UnityEngine.Quaternion, UnityEngine.Transform)
UnityEngine.Object.Instantiate(UnityEngine.Object)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Transform)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Transform, System.Boolean)
UnityEngine.Object.Instantiate<T>(T)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Vector3, UnityEngine.Quaternion)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Vector3, UnityEngine.Quaternion, UnityEngine.Transform)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Transform)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Transform, System.Boolean)
UnityEngine.Object.Destroy(UnityEngine.Object, System.Single)
UnityEngine.Object.Destroy(UnityEngine.Object)
UnityEngine.Object.DestroyImmediate(UnityEngine.Object, System.Boolean)
UnityEngine.Object.DestroyImmediate(UnityEngine.Object)
UnityEngine.Object.FindObjectsOfType(System.Type)
UnityEngine.Object.DontDestroyOnLoad(UnityEngine.Object)
UnityEngine.Object.DestroyObject(UnityEngine.Object, System.Single)
UnityEngine.Object.DestroyObject(UnityEngine.Object)
UnityEngine.Object.FindSceneObjectsOfType(System.Type)
UnityEngine.Object.FindObjectsOfTypeIncludingAssets(System.Type)
UnityEngine.Object.FindObjectsOfType<T>()
UnityEngine.Object.FindObjectOfType<T>()
UnityEngine.Object.FindObjectsOfTypeAll(System.Type)
UnityEngine.Object.FindObjectOfType(System.Type)
UnityEngine.Object.ToString()
UnityEngine.Object.name
UnityEngine.Object.hideFlags
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Stem

Assembly: Stem.dll

Syntax

```
[CreateAssetMenu(fileName = "New Music Bank", menuName = "Stem/Music Bank")]
public class MusicBank : ScriptableObject, ISerializationCallbackReceiver, IBank
```

Properties

Players

The collection of music players.

Declaration

```
public ReadOnlyCollection<MusicPlayer> Players { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.ObjectModel.ReadOnlyCollection<MusicPlayer>	A reference to a read-only collection of music players.

Playlists

The collection of playlists.

Declaration

```
public ReadOnlyCollection<Playlist> Playlists { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.ObjectModel.ReadOnlyCollection<Playlist>	A reference to a read-only collection of playlists.

ShowPlayers

The flag indicating whether the music bank inspector should show the 'Players' group.

Declaration

```
public bool ShowPlayers { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if the 'Players' group is shown. False otherwise.

Remarks

This property is used only by the music bank inspector and does nothing during runtime.

ShowPlaylists

The flag indicating whether the music bank inspector should show the 'Playlists' group.

Declaration

```
public bool ShowPlaylists { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if the 'Playlists' group is shown. False otherwise.

Remarks

This property is used only by the music bank inspector and does nothing during runtime.

Methods

AddMusicPlayer(String)

Adds a new music player to the music bank.

Declaration

```
public MusicPlayer AddMusicPlayer(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the music player.

Returns

TYPE	DESCRIPTION
MusicPlayer	A reference to a newly created music player.

AddPlaylist(String)

Adds a new playlist to the music bank.

Declaration

```
public Playlist AddPlaylist(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the playlist.

Returns

TYPE	DESCRIPTION
Playlist	A reference to a newly created playlist.

ContainsAsset(String)

Determines whether the music bank contains a music player with a matching name.

Declaration

```
public bool ContainsAsset(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the music player.

Returns

TYPE	DESCRIPTION
System.Boolean	True, if the music bank contains music player with a matching name. False otherwise.

GetMusicPlayer(String)

Searches for the specified music player with a matching name.

Declaration

```
public MusicPlayer GetMusicPlayer(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the music player.

Returns

TYPE	DESCRIPTION
MusicPlayer	A reference to a music player, if found. Null reference otherwise.

GetPlaylist(String)

Searches for the specified playlist with a matching name.

Declaration

```
public Playlist GetPlaylist(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the playlist.

Returns

TYPE	DESCRIPTION
Playlist	A reference to a playlist, if found. Null reference otherwise.

OnAfterDeserialize()

Prepares music bank for runtime use after deserialization.

Declaration

```
public void OnAfterDeserialize()
```

Remarks

This method is automatically called by Unity during deserialization process. Don't call it manually.

OnBeforeSerialize()

Prepares music bank for serialization.

Declaration

```
public void OnBeforeSerialize()
```

Remarks

This method is automatically called by Unity during serialization process. Don't call it manually.

RemoveMusicPlayer(MusicPlayer)

Removes existing music player from the music bank.

Declaration

```
public void RemoveMusicPlayer(MusicPlayer player)
```

Parameters

TYPE	NAME	DESCRIPTION
MusicPlayer	player	A reference to a music player.

Remarks

This method does nothing if the music player was not found in the music bank.

RemovePlaylist(Playlist)

Removes existing playlist from the music bank.

Declaration

```
public void RemovePlaylist(Playlist playlist)
```

Parameters

TYPE	NAME	DESCRIPTION
Playlist	playlist	A reference to a playlist.

Remarks

This method does nothing if the playlist was not found in the music bank.

All existing music players containing removed playlist will set their playlist reference to null.

Implements

UnityEngine.ISerializationCallbackReceiver

Class MusicManager

The main class for music playback and bank management.

Inheritance

System.Object
MusicManager

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Stem

Assembly: Stem.dll

Syntax

```
public static class MusicManager
```

Properties

Banks

The collection of all registered music banks.

Declaration

```
public static ReadOnlyCollection<MusicBank> Banks { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.ObjectModel.ReadOnlyCollection<MusicBank>	A reference to a read-only collection of music banks.

PrimaryBank

The primary music bank that will be searched first in case of name collisions.

Declaration

```
public static MusicBank PrimaryBank { get; set; }
```

Property Value

TYPE	DESCRIPTION
MusicBank	A reference to a primary music bank.

Methods

GetMusicPlayer(String)

Searches for the specified music player with a matching name.

Declaration

```
public static MusicPlayer GetMusicPlayer(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	

Returns

TYPE	DESCRIPTION
MusicPlayer	A reference to a music player, if found. Null reference otherwise.

Remarks

If multiple banks have music players with a matching name, primary music bank will be checked first. Within a bank, the first occurrence of found music player will be used.

Next(String)

Advances music player to next track.

Declaration

```
public static void Next(string player)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	player	Name of the music player.

Remarks

If multiple banks have music players with a matching name, primary music bank will be checked first. Within a bank, the first occurrence of found music player will be used.

This method does nothing if no playlist was assigned to the music player. Use SetPlaylist(String, String) or SetPlaylist(String, String, Single) to assign a playlist.

Next(String, Single)

Advances music player to next track.

Declaration

```
public static void Next(string player, float fade)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	player	Name of the music player.

TYPE	NAME	DESCRIPTION
System.Single	fade	Crossfade duration in seconds.

Remarks

If multiple banks have music players/playlists with a matching name, primary music bank will be checked first. Within a bank, the first occurrence of found music player/playlist will be used.

This method does nothing if no playlist was assigned to the music player. Use SetPlaylist(String, String) or SetPlaylist(String, String, Single) to assign a playlist.

Crossfade parameter value will override MusicPlayer.Fade value.

Pause(String)

Pauses music player.

Declaration

```
public static void Pause(string player)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	player	Name of the music player.

Remarks

If multiple banks have music players/playlists with a matching name, primary music bank will be checked first. Within a bank, the first occurrence of found music player/playlist will be used.

This method does nothing if no playlist was assigned to the music player. Use SetPlaylist(String, String) or SetPlaylist(String, String, Single) to assign a playlist.

Pause(String, Single)

Pauses music player.

Declaration

```
public static void Pause(string player, float fade)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	player	Name of the music player.
System.Single	fade	Crossfade duration in seconds.

Remarks

If multiple banks have music players/playlists with a matching name, primary music bank will be checked first. Within a bank, the

first occurrence of found music player/playlist will be used.

This method does nothing if no playlist was assigned to the music player. Use SetPlaylist(String, String) or SetPlaylist(String, String, Single) to assign a playlist.

Crossfade parameter value will override MusicPlayer.Fade value.

Play(String)

Plays music player.

Declaration

```
public static void Play(string player)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	player	Name of the music player.

Remarks

If multiple banks have music players/playlists with a matching name, primary music bank will be checked first. Within a bank, the first occurrence of found music player/playlist will be used.

This method does nothing if no playlist was assigned to the music player. Use SetPlaylist(String, String) or SetPlaylist(String, String, Single) to assign a playlist.

Play(String, Single)

Plays music player.

Declaration

```
public static void Play(string player, float fade)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	player	Name of the music player.
System.Single	fade	Crossfade duration in seconds.

Remarks

If multiple banks have music players/playlists with a matching name, primary music bank will be checked first. Within a bank, the first occurrence of found music player/playlist will be used.

This method does nothing if no playlist was assigned to the music player. Use SetPlaylist(String, String) or SetPlaylist(String, String, Single) to assign a playlist.

Crossfade parameter value will override MusicPlayer.Fade value.

Prev(String)

Advances music player to previous track.

Declaration

```
public static void Prev(string player)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	player	Name of the music player.

Remarks

If multiple banks have music players/playlists with a matching name, primary music bank will be checked first. Within a bank, the first occurrence of found music player/playlist will be used.

This method does nothing if no playlist was assigned to the music player. Use SetPlaylist(String, String) or SetPlaylist(String, String, Single) to assign a playlist.

Prev(String, Single)

Advances music player to previous track.

Declaration

```
public static void Prev(string player, float fade)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	player	Name of the music player.
System.Single	fade	Crossfade duration in seconds.

Remarks

If multiple banks have music players/playlists with a matching name, primary music bank will be checked first. Within a bank, the first occurrence of found music player/playlist will be used.

This method does nothing if no playlist was assigned to the music player. Use SetPlaylist(String, String) or SetPlaylist(String, String, Single) to assign a playlist.

Crossfade parameter value will override MusicPlayer.Fade value.

RegisterBank(MusicBank)

Registers new music bank.

Declaration

```
public static bool RegisterBank(MusicBank bank)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
MusicBank	bank	A reference to a music bank to register.

Returns

TYPE	DESCRIPTION
System.Boolean	True, if music bank was succesfully registered. False otherwise.

Seek(String, String)

Advances music player to a track with a matching name.

Declaration

```
public static void Seek(string player, string track)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	player	Name of the music player.
System.String	track	Name of the track.

Remarks

Target track must be one of current playlist tracks.

If multiple banks have music players/playlists with a matching name, primary music bank will be checked first. Within a bank, the first occurrence of found music player/playlist will be used.

This method does nothing if no playlist was assigned to the music player. Use SetPlaylist(String, String) or SetPlaylist(String, String, Single) to assign a playlist.

Seek(String, String, Single)

Advances music player to a track with a matching name.

Declaration

```
public static void Seek(string player, string track, float fade)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	player	Name of the music player.
System.String	track	Name of the track.

TYPE	NAME	DESCRIPTION
System.Single	fade	Crossfade duration in seconds.

Remarks

Target track must be one of current playlist tracks.

If multiple banks have music players/playlists with a matching name, primary music bank will be checked first. Within a bank, the first occurrence of found music player/playlist will be used.

This method does nothing if no playlist was assigned to the music player. Use SetPlaylist(String, String) or SetPlaylist(String, String, Single) to assign a playlist.

Crossfade parameter value will override MusicPlayer.Fade value.

SetPlaylist(String, String)

Sets a playlist to a music player.

Declaration

```
public static void SetPlaylist(string player, string playlist)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	player	Name of the music player.
System.String	playlist	Name of the playlist.

Remarks

If multiple banks have music players/playlists with a matching name, primary music bank will be checked first. Within a bank, the first occurrence of found music player/playlist will be used.

If music player was playing another track it'll automatically crossfade to the first track of the new playlist.

SetPlaylist(String, String, Single)

Sets a playlist to a music player.

Declaration

```
public static void SetPlaylist(string player, string playlist, float fade)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	player	Name of the music player.
System.String	playlist	Name of the playlist.

TYPE	NAME	DESCRIPTION
System.Single	fade	Crossfade duration in seconds.

Remarks

If multiple banks have music players/playlists with a matching name, primary music bank will be checked first. Within a bank, the first occurrence of found music player/playlist will be used.

If music player was playing another track it'll automatically crossfade to first track of the new playlist.

Crossfade parameter value will override MusicPlayer.Fade value.

Stop(String)

Stops music player.

Declaration

```
public static void Stop(string player)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	player	Name of the music player.

Remarks

If multiple banks have music players/playlists with a matching name, primary music bank will be checked first. Within a bank, the first occurrence of found music player/playlist will be used.

This method does nothing if no playlist was assigned to the music player. Use SetPlaylist(String, String) or SetPlaylist(String, String, Single) to assign a playlist.

Stop(String, Single)

Stops music player.

Declaration

```
public static void Stop(string player, float fade)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	player	Name of the music player.
System.Single	fade	Crossfade duration in seconds.

Remarks

If multiple banks have music players/playlists with a matching name, primary music bank will be checked first. Within a bank, the first occurrence of found music player/playlist will be used.

This method does nothing if no playlist was assigned to the music player. Use SetPlaylist(String, String) or SetPlaylist(String, String, Single) to assign a playlist.

Crossfade parameter value will override MusicPlayer.Fade value.

UnPause(String)

Resumes music player.

Declaration

```
public static void UnPause(string player)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	player	Name of the music player.

Remarks

If multiple banks have music players/playlists with a matching name, primary music bank will be checked first. Within a bank, the first occurrence of found music player/playlist will be used.

This method does nothing if no playlist was assigned to the music player. Use SetPlaylist(String, String) or SetPlaylist(String, String, Single) to assign a playlist.

UnPause(String, Single)

Resumes music player.

Declaration

```
public static void UnPause(string player, float fade)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	player	Name of the music player.
System.Single	fade	Crossfade duration in seconds.

Remarks

If multiple banks have music players/playlists with a matching name, primary music bank will be checked first. Within a bank, the first occurrence of found music player/playlist will be used.

This method does nothing if no playlist was assigned to the music player. Use SetPlaylist(String, String) or SetPlaylist(String, String, Single) to assign a playlist.

Crossfade parameter value will override MusicPlayer.Fade value.

Class MusicPlayer

The persistent storage for playback rules of a particular playlist.

Inheritance

System.Object
MusicPlayer

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Stem

Assembly: Stem.dll

Syntax

```
[Serializable]
public class MusicPlayer
```

Properties

Audible

The flag indicating if the music player can be heard.

Declaration

```
public bool Audible { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if the music player can be heard. False otherwise.

AutoAdvance

The flag indicating whether the music player should auto advance to next playlist track it finishes.

Declaration

```
public bool AutoAdvance { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if music player is auto advancing playlist tracks. False otherwise.

Remarks

This flag works in pair with Loop flag. There are four possible combinations:

1. Both flags false — play the current track once and then stop.
2. Both flags true — play all playlist tracks in a loop and never stop.
3. Loop is false, AutoAdvance is true — play all playlist tracks once and then stop.
4. Loop is true, AutoAdvance is false — play the current track in a loop and never stop.

Bank

The music bank the music player belongs to.

Declaration

```
public MusicBank Bank { get; set; }
```

Property Value

TYPE	DESCRIPTION
MusicBank	A reference to a music bank.

Fade

The crossfade parameter that is used when the music player transitions between tracks or playback states.

Declaration

```
public float Fade { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Single	Crossfade duration in seconds.

Loop

The flag indicating whether the music player should repeat playlist tracks after they finish.

Declaration

```
public bool Loop { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if the music player is looping playlist tracks. False otherwise.

Remarks

This flag works in pair with AutoAdvance flag. There are four possible combinations:

1. Both flags false — play the current track once and then stop.
2. Both flags true — play all playlist tracks in a loop and never stop.
3. Loop is false, AutoAdvance is true — play all playlist tracks once and then stop.
4. Loop is true, AutoAdvance is false — play the current track in a loop and never stop.

MixerGroup

The reference to an audio mixer group. Please refer to Unity Scripting Reference for details.

Declaration

```
public AudioMixerGroup MixerGroup { get; set; }
```

Property Value

TYPE	DESCRIPTION
UnityEngine.Audio.AudioMixerGroup	A reference to a mixer group.

Muted

The flag indicating if the music player is muted and can't be heard.

Declaration

```
public bool Muted { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if the music player is muted. False otherwise.

Remarks

This flag may be overridden by the Soloed flag, i.e. if the music player is simultaneously muted and soloed it'll be audible.

Name

The name of the music player. Used for fast search in corresponding music bank.

Declaration

```
public string Name { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.String	Name of the music player.

Playlist

The reference to a playlist which will be played.

Declaration

```
public Playlist Playlist { get; set; }
```

Property Value

TYPE	DESCRIPTION

TYPE	DESCRIPTION
Playlist	A reference to a playlist.

Shuffle

The flag indicating whether the music player should play tracks in random order.

Declaration

```
public bool Shuffle { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if the music player is playing playlist tracks in random order. False if it's playing sequentially.

Remarks

Note that tracks will be reshuffled again after the player will finish playing all the tracks.

Soloed

The flag indicating if the music player is soloed. If set to true, all other non-solo music players won't be audible.

Declaration

```
public bool Soloed { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if the music player is soloed. False otherwise.

Remarks

This flag may override the Muted flag, i.e. if the music player is simultaneously muted and soloed it'll be audible.

Unfolded

The flag indicating whether the music bank inspector should show advanced settings for the music player.

Declaration

```
public bool Unfolded { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if advanced settings are shown. False otherwise.

Remarks

This property is used only by the music bank inspector and does nothing during runtime.

Volume

The volume of the music player.

Declaration

```
public float Volume { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Single	Volume of the music player. Value must be in [0;1] range.

Class Playlist

The persistent collection of playlist tracks.

Inheritance

System.Object
Playlist

Implements

UnityEngine.ISerializationCallbackReceiver

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Stem

Assembly: Stem.dll

Syntax

```
[Serializable]
public class Playlist : ISerializationCallbackReceiver
```

Properties

Bank

The music bank the playlist belongs to.

Declaration

```
public MusicBank Bank { get; set; }
```

Property Value

TYPE	DESCRIPTION
MusicBank	A reference to a music bank.

Name

The name of the playlist. Used for fast search in corresponding music bank.

Declaration

```
public string Name { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.String	Name of the playlist.

Tracks

The collection of playlist tracks.

Declaration

```
public ReadOnlyCollection<PlaylistTrack> Tracks { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.ObjectModel.ReadOnlyCollection<PlaylistTrack>	A reference to a read-only collection of playlist tracks.

Unfolded

The flag indicating whether the music bank inspector should show advanced settings for the playlist.

Declaration

```
public bool Unfolded { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if advanced settings are shown. False otherwise.

Remarks

This property is used only by the music bank inspector and does nothing during runtime.

Methods

OnAfterDeserialize()

Prepares playlist for runtime use after deserialization.

Declaration

```
public void OnAfterDeserialize()
```

Remarks

This method is automatically called by Unity during deserialization process. Don't call it manually.

OnBeforeSerialize()

Prepares playlist for serialization.

Declaration

```
public void OnBeforeSerialize()
```

Remarks

This method is automatically called by Unity during serialization process. Don't call it manually.

Implements

UnityEngine.ISerializationCallbackReceiver

Class PlaylistTrack

The persistent storage for music audio data.

Inheritance

System.Object
PlaylistTrack

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Stem

Assembly: Stem.dll

Syntax

```
[Serializable]
public class PlaylistTrack
```

Properties

Clip

The audio clip with audio data. Please refer to Unity Scripting Reference for details.

Declaration

```
public AudioClip Clip { get; set; }
```

Property Value

TYPE	DESCRIPTION
UnityEngine.AudioClip	A reference to an audio clip.

Name

The name of the track.

Declaration

```
public string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	Name of the current audio clip being used. Null reference otherwise.

Playlist

The playlist the track belongs to.

Declaration

```
public Playlist Playlist { get; set; }
```

Property Value

TYPE	DESCRIPTION
Playlist	A reference to a playlist.

Volume

The volume of the track.

Declaration

```
public float Volume { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Single	Volume of the track. Value must be in [0;1] range.

Enum RetriggerMode

Defines how sound will play its variations.

Namespace: Stem

Assembly: Stem.dll

Syntax

```
[Serializable]
public enum RetriggerMode
```

Fields

NAME	DESCRIPTION
Random	Play random variations with possible repetitions.
RandomNoRepeat	Play random variations without repetitions.
Sequential	Play variations in a sequence as they stored in the sound.

Class Sound

The persistent storage for sound variations and the most important audio source settings.

Inheritance

System.Object
Sound

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Stem

Assembly: Stem.dll

Syntax

```
[Serializable]
public class Sound
```

Properties

AttenuationMode

The attenuation mode defining how sound volume will be lowered over the distance.

Declaration

```
public AttenuationMode AttenuationMode { get; set; }
```

Property Value

TYPE	DESCRIPTION
AttenuationMode	An enum value.

Remarks

Note that attenuation rules applies only for 3D sounds. Please refer to Unity Scripting Reference for details.

Audible

The flag indicating if the sound can be heard.

Declaration

```
public bool Audible { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if the sound can be heard. False otherwise.

Remarks

If the Bus is inaudible, the sound will also be inaudible.

Bank

The sound bank the sound belongs to.

Declaration

```
public SoundBank Bank { get; set; }
```

Property Value

TYPE	DESCRIPTION
SoundBank	A reference to a sound bank.

Bus

The reference to a sound bus which will manage the sound.

Declaration

```
public SoundBus Bus { get; set; }
```

Property Value

TYPE	DESCRIPTION
SoundBus	A reference to a sound bus.

Remarks

If set to null, DefaultBus will be used.

DopplerLevel

The parameter defining the Doppler scale for the sound.

Declaration

```
public float DopplerLevel { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Single	Doppler scale of the sound. The value must be greater or equal to zero.

Remarks

This parameter duplicates corresponding parameter from AudioSource. Please refer to Unity Scripting Reference for details.

MaxDistance

The parameter defining the boundary outside which the sound will be inaudible or stop attenuating depending on AttenuationMode value.

Declaration

```
public float MaxDistance { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Single	Distance in units.

Remarks

This parameter duplicates corresponding parameter from AudioSource. Please refer to Unity Scripting Reference for details.

MinDistance

The parameter defining the boundary within which the sound won't get any louder.

Declaration

```
public float MinDistance { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Single	Distance in units.

Remarks

This parameter duplicates corresponding parameter from AudioSource. Please refer to Unity Scripting Reference for details.

Muted

The flag indicating if the sound is muted and can't be heard.

Declaration

```
public bool Muted { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if the sound is muted. False otherwise.

Remarks

This flag may be overridden by the Soloed flag, i.e. if the sound is simultaneously muted and soloed it'll be audible.

Name

The name of the sound. Used for fast search in corresponding sound bank.

Declaration

```
public string Name { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.String	Name of the sound.

PanStereo

The stereo panning parameter defining sound position in a stereo way (left or right).

Declaration

```
public float PanStereo { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Single	Stereo pan of the sound. The value must be in [-1;1] range.

Remarks

This parameter duplicates corresponding parameter from AudioSource. Please refer to Unity Scripting Reference for details.

Soloed

The flag indicating if the sound is soloed. If set to true, all other non-solo sounds won't be audible.

Declaration

```
public bool Soloed { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if the sound is soloed. False otherwise.

Remarks

This flag may override the Muted flag, i.e. if the sound is simultaneously muted and soloed it'll be audible.

SpatialBlend

The spatial blend parameter defining how much the sound is affected by 3d spatialisation calculations (attenuation, doppler etc).

Declaration

```
public float SpatialBlend { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Single	Spatial blend of the sound. The value must be in [0;1] range.

Remarks

This parameter duplicates corresponding parameter from AudioSource. Please refer to Unity Scripting Reference for details.

Spread

The parameter defining the spread angle (in degrees) of a 3d stereo or multichannel sound in speaker space.

Declaration

```
public float Spread { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Single	Spread angle of the sound. The value must be in [0;360] range.

Remarks

This parameter duplicates corresponding parameter from AudioSource. Please refer to Unity Scripting Reference for details.

Unfolded

The flag indicating whether the sound bank inspector should show advanced settings for the sound.

Declaration

```
public bool Unfolded { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if advanced settings are shown. False otherwise.

Remarks

This property is used only by the sound bank inspector and does nothing during runtime.

VariationRetriggerMode

The retrigger mode defining how the sound will play variations.

Declaration

```
public RetriggerMode VariationRetriggerMode { get; set; }
```

Property Value

TYPE	DESCRIPTION
RetriggerMode	An enum value.

Variations

The collection of sound variations.

Declaration

```
public ReadOnlyCollection<SoundVariation> Variations { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.ObjectModel.ReadOnlyCollection<SoundVariation>	A reference to a read-only collection of sound variations.

Class SoundBank

The persistent storage for sounds and sound buses.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.ScriptableObject
SoundBank

Implements

UnityEngine.ISerializationCallbackReceiver

Inherited Members

UnityEngine.ScriptableObject.SetDirty()
UnityEngine.ScriptableObject.CreateInstance(System.String)
UnityEngine.ScriptableObject.CreateInstance(System.Type)
UnityEngine.ScriptableObject.CreateInstance<T>()
UnityEngine.Object.GetInstanceID()
UnityEngine.Object.GetHashCode()
UnityEngine.Object.Equals(System.Object)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Vector3, UnityEngine.Quaternion)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Vector3, UnityEngine.Quaternion, UnityEngine.Transform)
UnityEngine.Object.Instantiate(UnityEngine.Object)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Transform)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Transform, System.Boolean)
UnityEngine.Object.Instantiate<T>(T)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Vector3, UnityEngine.Quaternion)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Vector3, UnityEngine.Quaternion, UnityEngine.Transform)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Transform)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Transform, System.Boolean)
UnityEngine.Object.Destroy(UnityEngine.Object, System.Single)
UnityEngine.Object.Destroy(UnityEngine.Object)
UnityEngine.Object.DestroyImmediate(UnityEngine.Object, System.Boolean)
UnityEngine.Object.DestroyImmediate(UnityEngine.Object)
UnityEngine.Object.FindObjectsOfType(System.Type)
UnityEngine.Object.DontDestroyOnLoad(UnityEngine.Object)
UnityEngine.Object.DestroyObject(UnityEngine.Object, System.Single)
UnityEngine.Object.DestroyObject(UnityEngine.Object)
UnityEngine.Object.FindSceneObjectsOfType(System.Type)
UnityEngine.Object.FindObjectsOfTypeIncludingAssets(System.Type)
UnityEngine.Object.FindObjectsOfType<T>()
UnityEngine.Object.FindObjectOfType<T>()
UnityEngine.Object.FindObjectsOfTypeAll(System.Type)
UnityEngine.Object.FindObjectOfType(System.Type)
UnityEngine.Object.ToString()
UnityEngine.Object.name
UnityEngine.Object.hideFlags
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Stem

Assembly: Stem.dll

Syntax

```
[CreateAssetMenu(fileName = "New Sound Bank", menuName = "Stem/Sound Bank")]
public class SoundBank : ScriptableObject, ISerializationCallbackReceiver, IBank
```

Properties

Buses

The collection of sound buses.

Declaration

```
public ReadOnlyCollection<SoundBus> Buses { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.ObjectModel.ReadOnlyCollection<SoundBus>	A reference to a read-only collection of sound buses.

DefaultBus

The sound bus which will be used by default on newly created sounds.

Declaration

```
public SoundBus DefaultBus { get; }
```

Property Value

TYPE	DESCRIPTION
SoundBus	A reference to a sound bus.

ShowSoundBuses

The flag indicating whether the sound bank inspector should show the 'Buses' group.

Declaration

```
public bool ShowSoundBuses { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if the 'Buses' group is shown. False otherwise.

Remarks

This property is used only by the sound bank inspector and does nothing during runtime.

ShowSounds

The flag indicating whether the sound bank inspector should show the 'Sounds' group.

Declaration

```
public bool ShowSounds { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if the 'Sounds' group is shown. False otherwise.

Remarks

This property is used only by the sound bank inspector and does nothing during runtime.

SoundBatchImportMode

The batch import mode defining how new sounds will be created after the drag-drop event.

Declaration

```
public SoundBatchImportMode SoundBatchImportMode { get; set; }
```

Property Value

TYPE	DESCRIPTION
SoundBatchImportMode	An enum value.

Sounds

The collection of sounds.

Declaration

```
public ReadOnlyCollection<Sound> Sounds { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.ObjectModel.ReadOnlyCollection<Sound>	A reference to a read-only collection of sounds.

Methods

AddSound(String)

Adds an empty sound to the sound bank.

Declaration

```
public Sound AddSound(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the sound.

Returns

TYPE	DESCRIPTION
Sound	A reference to a newly created sound.

AddSound(String, AudioClip)

Adds a sound with a single variation to the sound bank.

Declaration

```
public Sound AddSound(string name, AudioClip clip)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the sound.
UnityEngine.AudioClip	clip	A reference to the audio clip with audio data.

Returns

TYPE	DESCRIPTION
Sound	A reference to a newly created sound.

AddSound(String, AudioClip[])

Adds a sound with multiple variations to the sound bank.

Declaration

```
public Sound AddSound(string name, AudioClip[] clips)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the sound.
UnityEngine.AudioClip[]	clips	An array of audio clips with audio data.

Returns

TYPE	DESCRIPTION
Sound	A reference to a newly created sound.

AddSoundBus(String)

Adds a new sound bus to the sound bank.

Declaration

```
public SoundBus AddSoundBus(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the sound bus.

Returns

TYPE	DESCRIPTION
SoundBus	A reference to a newly created sound bus.

ContainsAsset(String)

Determines whether the sound bank contains a sound with a matching name.

Declaration

```
public bool ContainsAsset(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the sound.

Returns

TYPE	DESCRIPTION
System.Boolean	True, if the sound bank contains sound with a matching name. False otherwise.

GetSound(String)

Searches for the specified sound with a matching name.

Declaration

```
public Sound GetSound(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the sound.

Returns

TYPE	DESCRIPTION
Sound	A reference to a sound, if found. Null reference otherwise.

GetSoundBus(String)

Searches for the specified sound bus with a matching name.

Declaration

```
public SoundBus GetSoundBus(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the sound bus.

Returns

TYPE	DESCRIPTION
SoundBus	A reference to a sound bus, if found. Null reference otherwise.

OnAfterDeserialize()

Prepares sound bank for runtime use after deserialization.

Declaration

```
public void OnAfterDeserialize()
```

Remarks

This method is automatically called by Unity during deserialization process. Don't call it manually.

OnBeforeSerialize()

Prepares sound bank for serialization.

Declaration

```
public void OnBeforeSerialize()
```

Remarks

This method is automatically called by Unity during serialization process. Don't call it manually.

RemoveSound(Sound)

Removes existing sound from the sound bank.

Declaration

```
public void RemoveSound(Sound sound)
```

Parameters

TYPE	NAME	DESCRIPTION
Sound	sound	A reference to a sound.

Remarks

This method does nothing if the sound was not found in the sound bank.

RemoveSoundBus(SoundBus)

Removes existing sound bus from the sound bank.

Declaration

```
public void RemoveSoundBus(SoundBus bus)
```

Parameters

TYPE	NAME	DESCRIPTION
SoundBus	bus	A reference to a sound bus.

Remarks

This method does nothing if the sound bus was not found in the sound bank.

Implements

UnityEngine.ISerializationCallbackReceiver

Enum SoundBatchImportMode

Defines how new sounds will be created after the drag-drop event.

Namespace: Stem

Assembly: Stem.dll

Syntax

```
public enum SoundBatchImportMode
```

Fields

NAME	DESCRIPTION
MultiplePerClip	A sound with a single variation will be created per each provided audio clip.
SingleWithVariations	A single sound will be created. Provided audio clips will be used as variations.

Class SoundBus

The persistent storage for playback rules of a group of sounds.

Inheritance

System.Object
SoundBus

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Stem

Assembly: Stem.dll

Syntax

```
[Serializable]
public class SoundBus
```

Properties

AllowVoiceStealing

The flag indicating whether the sound bus can stop the oldest playing sound and play the new one if Polyphony limit is exceeded.

Declaration

```
public bool AllowVoiceStealing { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if sound bus will be stopping the oldest playing sounds. False if the sound bus won't allow new sounds to play.

Audible

The flag indicating if the sound bus can be heard.

Declaration

```
public bool Audible { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if the sound bus can be heard. False otherwise.

Bank

The sound bank the sound bus belongs to.

Declaration

```
public SoundBank Bank { get; set; }
```

Property Value

TYPE	DESCRIPTION
SoundBank	A reference to a sound bank.

MixerGroup

The reference to an audio mixer group. Please refer to Unity Scripting Reference for details.

Declaration

```
public AudioMixerGroup MixerGroup { get; set; }
```

Property Value

TYPE	DESCRIPTION
UnityEngine.Audio.AudioMixerGroup	A reference to a mixer group.

Muted

The flag indicating if the sound bus is muted and can't be heard.

Declaration

```
public bool Muted { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if the sound bus is muted. False otherwise.

Remarks

This flag may be overridden by the Soloed flag, i.e. if the sound bus is simultaneously muted and soloed it'll be audible.

Name

The name of the sound bus. Used for fast search in corresponding sound bank.

Declaration

```
public string Name { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.String	Name of the sound bus.

Polyphony

The number of maximum allowed simultaneously playing sounds in the sound bus.

Declaration

```
public byte Polyphony { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Byte	Number of allowed sounds.

Soloed

The flag indicating if the sound bus is soloed. If set to true, all other non-solo sound buses won't be audible.

Declaration

```
public bool Soloed { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if the sound bus is soloed. False otherwise.

Remarks

This flag may override the Muted flag, i.e. if the sound bus is simultaneously muted and soloed it'll be audible.

Unfolded

The flag indicating whether the sound bank inspector should show advanced settings for the sound bus.

Declaration

```
public bool Unfolded { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if advanced settings are shown. False otherwise.

Remarks

This property is used only by the sound bank inspector and does nothing during runtime.

Volume

The volume of the sound bus.

Declaration

```
public float Volume { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Single	Volume of the sound bus. Value must be in [0;1] range.

Class SoundInstance

The game object with audio source component. Used for manual playback and custom mixing logic.

Inheritance

System.Object
SoundInstance

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Stem

Assembly: Stem.dll

Syntax

```
public class SoundInstance
```

Properties

Looped

The flag indicating that the sound instance is looping. Set whether it should replay the audio clip after it finishes.

Declaration

```
public bool Looped { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if sound instance is looping. False otherwise.

Paused

The flag indicating that the sound instance is paused.

Declaration

```
public bool Paused { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if sound instance is paused. False otherwise.

Pitch

The pitch property allows controlling how high or low the tone of the audio source is.

Declaration

```
public float Pitch { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Single	Pitch value in [3;3] range.

Playing

The flag indicating that the sound instance is playing.

Declaration

```
public bool Playing { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if sound instance is playing. False otherwise.

Sound

The reference to a sound which will be used for playback. Changing this value allows playing different sounds.

Declaration

```
public Sound Sound { get; set; }
```

Property Value

TYPE	DESCRIPTION
Sound	A reference to a sound.

TimeSamples

The playback position in samples.

Declaration

```
public int TimeSamples { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	An offset in samples from the start of an audio clip.

Transform

The transform component from the game object of the sound instance. Please refer to Unity Scripting Reference for details.

Declaration

```
public Transform Transform { get; }
```

Property Value

TYPE	DESCRIPTION
UnityEngine.Transform	A reference to a transform component.

Volume

The volume property allows controlling the overall level of sound coming to the audio source.

Declaration

```
public float Volume { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Single	Volume value in [0;1] range.

Methods

Pause()

Pauses sound.

Declaration

```
public void Pause()
```

Play()

Plays sound.

Declaration

```
public void Play()
```

Play(Single)

Plays sound.

Declaration

```
public void Play(float volume)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Single	volume	Volume of the sound. Value must be in [0;1] range.

Remarks

Volume parameter value will override SoundVariation.Volume value.

Play(Single, Single)

Plays sound.

Declaration

```
public void Play(float volume, float pitch)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Single	volume	Volume of the sound. Value must be in [0;1] range.
System.Single	pitch	Pitch of the sound. Value must be in [-3;3] range.

Remarks

Volume parameter value will override SoundVariation.Volume value.

Pitch parameter value will override SoundVariation.Pitch value.

Play(Single, Single, Single)

Plays sound.

Declaration

```
public void Play(float volume, float pitch, float delay)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Single	volume	Volume of the sound. Value must be in [0;1] range.
System.Single	pitch	Pitch of the sound. Value must be in [-3;3] range.
System.Single	delay	Delay of the sound. Value must be greater or equal to zero.

Remarks

Volume parameter value will override SoundVariation.Volume value.

Pitch parameter value will override SoundVariation.Pitch value.

Delay parameter value will override SoundVariation.Delay value.

Play3D(Vector3)

Plays sound in 3D space.

Declaration

```
public void Play3D(Vector3 position)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.Vector3	position	Position of the sound.

Play3D(Vector3, Single)

Plays sound in 3D space.

Declaration

```
public void Play3D(Vector3 position, float volume)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.Vector3	position	Position of the sound.
System.Single	volume	Volume of the sound. Value must be in [0;1] range.

Remarks

Volume parameter value will override SoundVariation.Volume value.

Play3D(Vector3, Single, Single)

Plays sound in 3D space.

Declaration

```
public void Play3D(Vector3 position, float volume, float pitch)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.Vector3	position	Position of the sound.
System.Single	volume	Volume of the sound. Value must be in [0;1] range.
System.Single	pitch	Pitch of the sound. Value must be in [-3;3] range.

Remarks

Volume parameter value will override SoundVariation.Volume value.

Pitch parameter value will override SoundVariation.Pitch value.

Play3D(Vector3, Single, Single, Single)

Plays sound in 3D space.

Declaration

```
public void Play3D(Vector3 position, float volume, float pitch, float delay)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.Vector3	position	Position of the sound.
System.Single	volume	Volume of the sound. Value must be in [0;1] range.
System.Single	pitch	Pitch of the sound. Value must be in [-3;3] range.
System.Single	delay	Delay of the sound. Value must be greater or equal to zero.

Remarks

Volume parameter value will override SoundVariation.Volume value.

Pitch parameter value will override SoundVariation.Pitch value.

Delay parameter value will override SoundVariation.Delay value.

Stop()

Stops sound.

Declaration

```
public void Stop()
```

UnPause()

Resumes sound.

Declaration

```
public void UnPause()
```

Class SoundManager

The main class for sound playback and bank management.

Inheritance

System.Object
SoundManager

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Stem

Assembly: Stem.dll

Syntax

```
public static class SoundManager
```

Properties

Banks

The collection of all registered sound banks.

Declaration

```
public static ReadOnlyCollection<SoundBank> Banks { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.ObjectModel.ReadOnlyCollection<SoundBank>	A reference to a read-only collection of sound banks.

PrimaryBank

The primary sound bank that will be searched first in case of name collisions.

Declaration

```
public static SoundBank PrimaryBank { get; set; }
```

Property Value

TYPE	DESCRIPTION
SoundBank	A reference to a primary sound bank.

Methods

GetSound(String)

Searches for the specified sound with a matching name.

Declaration

```
public static Sound GetSound(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	

Returns

TYPE	DESCRIPTION
Sound	A reference to a sound, if found. Null reference otherwise.

Remarks

If multiple banks have sounds with a matching name, primary sound bank will be checked first. Within a bank, the first occurrence of found sound will be used.

GrabSound()

Grabs an empty sound instance from the sound pool. Used for manual playback and custom mixing logic.

Declaration

```
public static SoundInstance GrabSound()
```

Returns

TYPE	DESCRIPTION
SoundInstance	A reference to an empty sound instance.

Remarks

This method may increase the size of the sound pool causing additional memory allocations.

When a sound instance is not needed anymore, use ReleaseSound(SoundInstance) to return it back to the sound pool.

GrabSound(String)

Grabs a sound instance from the sound pool. Used for manual playback and custom mixing logic.

Declaration

```
public static SoundInstance GrabSound(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	

Returns

TYPE	DESCRIPTION
SoundInstance	A reference to a sound instance.

Remarks

If multiple banks have sounds with a matching name, primary sound bank will be checked first. Within a bank, the first occurrence of found sound will be used.

This method may increase the size of the sound pool causing additional memory allocations.

When a sound instance is not needed anymore, use ReleaseSound(SoundInstance) to return it back to the sound pool.

Pause()

Pauses all playing sounds.

Declaration

```
public static void Pause()
```

Remarks

This method will also stop all sounds instances returned from GrabSound() or GrabSound(String).

Play(String)

Plays one-shot sound.

Declaration

```
public static void Play(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the sound.

Remarks

If multiple banks have sounds with a matching name, primary sound bank will be checked first. Within a bank, the first occurrence of found sound will be used.

Play(String, Single)

Plays one-shot sound.

Declaration

```
public static void Play(string name, float volume)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the sound.

TYPE	NAME	DESCRIPTION
System.Single	volume	Volume of the sound. Value must be in [0;1] range.

Remarks

If multiple banks have sounds with a matching name, primary sound bank will be checked first. Within a bank, the first occurrence of found sound will be used.

Volume parameter value will override SoundVariation.Volume value.

Play(String, Single, Single)

Plays one-shot sound.

Declaration

```
public static void Play(string name, float volume, float pitch)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the sound.
System.Single	volume	Volume of the sound. Value must be in [0;1] range.
System.Single	pitch	Pitch of the sound. Value must be in [-3;3] range.

Remarks

If multiple banks have sounds with a matching name, primary sound bank will be checked first. Within a bank, the first occurrence of found sound will be used.

Volume parameter value will override SoundVariation.Volume value.

Pitch parameter value will override SoundVariation.Pitch value.

Play(String, Single, Single, Single)

Plays one-shot sound.

Declaration

```
public static void Play(string name, float volume, float pitch, float delay)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the sound.
System.Single	volume	Volume of the sound. Value must be in [0;1] range.

TYPE	NAME	DESCRIPTION
System.Single	pitch	Pitch of the sound. Value must be in [-3;3] range.
System.Single	delay	Delay of the sound. Value must be greater or equal to zero.

Remarks

If multiple banks have sounds with a matching name, primary sound bank will be checked first. Within a bank, the first occurrence of found sound will be used.

Volume parameter value will override SoundVariation.Volume value.

Pitch parameter value will override SoundVariation.Pitch value.

Delay parameter value will override SoundVariation.Delay value.

Play3D(String, Vector3)

Plays one-shot sound in 3D space.

Declaration

```
public static void Play3D(string name, Vector3 position)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the sound.
UnityEngine.Vector3	position	Position of the sound.

Remarks

If multiple banks have sounds with a matching name, primary sound bank will be checked first. Within a bank, the first occurrence of found sound will be used.

Play3D(String, Vector3, Single)

Plays one-shot sound in 3D space.

Declaration

```
public static void Play3D(string name, Vector3 position, float volume)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
System.String	name	Name of the sound.
UnityEngine.Vector3	position	Position of the sound.
System.Single	volume	Volume of the sound. Value must be in [0;1] range.

Remarks

If multiple banks have sounds with a matching name, primary sound bank will be checked first. Within a bank, the first occurrence of found sound will be used.

Volume parameter value will override SoundVariation.Volume value.

Play3D(String, Vector3, Single, Single)

Plays one-shot sound in 3D space.

Declaration

```
public static void Play3D(string name, Vector3 position, float volume, float pitch)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the sound.
UnityEngine.Vector3	position	Position of the sound.
System.Single	volume	Volume of the sound. Value must be in [0;1] range.
System.Single	pitch	Pitch of the sound. Value must be in [-3;3] range.

Remarks

If multiple banks have sounds with a matching name, primary sound bank will be checked first. Within a bank, the first occurrence of found sound will be used.

Volume parameter value will override SoundVariation.Volume value.

Pitch parameter value will override SoundVariation.Pitch value.

Play3D(String, Vector3, Single, Single, Single)

Plays one-shot sound in 3D space.

Declaration

```
public static void Play3D(string name, Vector3 position, float volume, float pitch, float delay)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Name of the sound.
UnityEngine.Vector3	position	Position of the sound.
System.Single	volume	Volume of the sound. Value must be in [0;1] range.
System.Single	pitch	Pitch of the sound. Value must be in [-3;3] range.
System.Single	delay	Delay of the sound. Value must be greater or equal to zero.

Remarks

If multiple banks have sounds with a matching name, primary sound bank will be checked first. Within a bank, the first occurrence of found sound will be used.

Volume parameter value will override SoundVariation.Volume value.

Pitch parameter value will override SoundVariation.Pitch value.

Delay parameter value will override SoundVariation.Delay value.

RegisterBank(SoundBank)

Registers new sound bank.

Declaration

```
public static bool RegisterBank(SoundBank bank)
```

Parameters

TYPE	NAME	DESCRIPTION
SoundBank	bank	A reference to a sound bank to register.

Returns

TYPE	DESCRIPTION
System.Boolean	True, if sound bank was succesfully registered. False otherwise.

ReleaseSound(SoundInstance)

Releases sound instance and return it back to the sound pool.

Declaration

```
public static bool ReleaseSound(SoundInstance instance)
```

Parameters

TYPE	NAME	DESCRIPTION
SoundInstance	instance	A reference to a sound instance.

Returns

TYPE	DESCRIPTION
System.Boolean	True, if the sound instance was successfully returned to sound pool. False otherwise.

Remarks

Once the sound instance is returned back to a sound pool, it's possible to reuse it again by calling GrabSound() or GrabSound(String).

Stop()

Stops all playing sounds.

Declaration

```
public static void Stop()
```

Remarks

This method will also stop all sounds instances returned from GrabSound() or GrabSound(String).

UnPause()

Resumes all paused sounds.

Declaration

```
public static void UnPause()
```

Remarks

This method will also resume all sounds instances returned from GrabSound() or GrabSound(String).

Class SoundVariation

The persistent storage for sound effect audio data.

Inheritance

System.Object
SoundVariation

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: Stem

Assembly: Stem.dll

Syntax

```
[Serializable]
public class SoundVariation
```

Properties

Clip

The audio clip with audio data. Please refer to Unity Scripting Reference for details.

Declaration

```
public AudioClip Clip { get; set; }
```

Property Value

TYPE	DESCRIPTION
UnityEngine.AudioClip	A reference to an audio clip.

Delay

The delay of the sound variation.

Declaration

```
public float Delay { get; }
```

Property Value

TYPE	DESCRIPTION
System.Single	FixedDelay value if RandomizeDelay is false. Otherwise random delay value from RandomDelay range.

FixedDelay

The fixed delay of the sound variation.

Declaration

```
public float FixedDelay { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Single	Fixed delay in seconds of the sound variation. Value must be greater or equal to zero.

FixedPitch

The fixed pitch of the sound variation.

Declaration

```
public float FixedPitch { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Single	Fixed pitch of the souhnd variation. Value must be in [-3;3] range.

FixedVolume

The fixed volume of the sound variation.

Declaration

```
public float FixedVolume { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Single	Fixed volume of the sound variation. Value must be in [0;1] range.

Name

The name of the sound variation.

Declaration

```
public string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	Name of the current audio clip being used. Null reference otherwise.

Pitch

The pitch of the sound variation.

Declaration

```
public float Pitch { get; }
```

Property Value

TYPE	DESCRIPTION
System.Single	FixedPitch value if RandomizePitch is false. Otherwise random pitch value from RandomPitch range.

RandomDelay

The random delay range of the sound variation.

Declaration

```
public Vector2 RandomDelay { get; set; }
```

Property Value

TYPE	DESCRIPTION
UnityEngine.Vector2	Random delay range of the sound variation. Vector components store range boundaries (x - min, y - max).

RandomizeDelay

The flag indicating which property is used for Delay.

Declaration

```
public bool RandomizeDelay { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if RandomDelay is used. Otherwise FixedDelay is used.

RandomizePitch

The flag indicating which property is used for Pitch.

Declaration

```
public bool RandomizePitch { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if RandomPitch is used. Otherwise FixedPitch is used.

RandomizeVolume

The flag indicating which property is used for Volume.

Declaration

```
public bool RandomizeVolume { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	True, if RandomVolume is used. Otherwise FixedVolume is used.

RandomPitch

The random pitch range of the sound variation.

Declaration

```
public Vector2 RandomPitch { get; set; }
```

Property Value

TYPE	DESCRIPTION
UnityEngine.Vector2	Random pitch range of the sound variation. Vector components store range boundaries (x - min, y - max).

RandomVolume

The random volume range of the sound variation.

Declaration

```
public Vector2 RandomVolume { get; set; }
```

Property Value

TYPE	DESCRIPTION
UnityEngine.Vector2	Random volume range of the sound variation. Vector components store range boundaries (x - min, y - max).

Volume

The volume of the sound variation.

Declaration

```
public float Volume { get; }
```

Property Value

TYPE	DESCRIPTION
System.Single	FixedVolume value if RandomizeVolume is false. Otherwise random volume value from RandomVolume range.