AUDIO

MANUAL

Table of Contents

Questions or problems?

support@makecodenow.com

Introduction

SECTR AUDIO brings the latest, cutting edge audio production tools and technologies to Unity, including an unparalleled suite of editor extensions and runtime components that let you create rich, complex soundscapes with ease and play them back with a minimum of CPU overhead. Of course, SECTR AUDIO also includes everything you expect from a quality audio production environment, including randomization, templates, hierarchical mixing, comprehensive asset management, and version control integration.

Crafting great interactive audio isn't just about making audio files, but about how, when, and where you play them. SECTR AUDIO gives designers a unique library of components for audio playback including volumetric sound sources, randomized audio environments, realtime occlusion, and dynamic sound propagation.

Audio designers know that mixing is one of the most difficult and time consuming parts of creating a great aural experience. In addition to hierarchical mixing, SECTR AUDIO also includes high dynamic range (HDR) mixing. When enabled, HDR mixing lets you play sounds that mimic the real world, with a huge range of loudnesses, all dynamically mixed down into an LDR range.

To make all of these features possible on everything from smartphones to PCs, SECTR AUDIO is written with performance in mind. All major sound allocations are pooled, and expensive calculations are time-sliced. SECTR AUDIO also includes a virtual sound system, intelligently culling sounds that are too quiet or too far away to be audible.

All SECTR modules include complete source code, online support, and are fully compatible with Unity, Unity Pro, and Unity Version Control.

Overview

SECTR AUDIO has three major aspects: creating sounds, playing them back, and mixing them together, each of which will be discussed briefly here, and at greater length below. Although part of the suite of SECTR products, SECTR AUDIO does not require any Sector or Portals, except to enable a couple of advanced features.

Creating Sounds

SECTR AUDIO introduces a new audio asset, the AudioCue. An AudioCue is the most basic unit of playable sound in SECTR AUDIO. AudioCues contain many of the same properties as a regular Unity AudioSource, but include many features beyond those basics, including randomization, fading, and all of the properties for advanced features like occlusion, propagation, etc.

Playing Sounds

All sounds in SECTR AUDIO route through the AudioSystem component. The AudioSystem provides a centralized interface for all sound playback and does all of the heavy lifting required to create new sounds, mix them, etc. Programmers can call this API directly, or use any of the included SECTR AUDIOSource components.

Mixing

SECTR AUDIO includes a standardize hierarchical mixer that should be familiar to most sound designers. Each AudioCue in the game has a parent bus, and its final volume is determined by the volume of its parent and all of its parents' parents. This hierarchy makes it easy to make both very broad and very specific mixing changes easily.

SECTR AUDIO also includes an optional, high dynamic range (HDR) mixer. Individual Cues can be tagged as HDR. When tagged this way, they are given a physically accurate representation of volume – called loudness, and the final volume is determined on the fly based on all of the other HDR sounds that are currently playing.

SECTR AUDIO Window

One of the biggest challenges in making game audio is simply managing all of the assets involved. SECTR AUDIO makes this task as easy as possible with the SECTR AUDIO window, which you can find under Window->SECTR->Audio. The SECTR AUDIO window has three panes, each of which will be explained below.

The SECTR AUDIO window supports drag and drop between panels. Dragging an AudioClip onto an AudioCue will add it to that AudioCue. Dragging it onto a bus will make a new AudioCue with that AudioClip in it. You can also drag AudioBuses and AudioCues around reparent them and otherwise move things around.

The SECTR AUDIO window is also version control aware. Any objects that are not locally editable will be displayed in gray with their controls disabled. You can right-click on any asset to check it out, or you can check it out through the normal Unity GUIs.

Audio Clip Panel

Audio Clips are the basis of of all playback in Unity, and SECTR AUDIO is no exception. Because a large game can have many AudioClips, the clip panel tries to make managing them as easy as possible. Clips are sorted by the folders they are contained in, and the list can be filtered using the search interface at the top. By default, only basic information is shown for each clip, but you can show all of the details by right clicking in the pane and choosing Show Full Details.

Note that in very large projects the sheer number of AudioClips can overwhelm Unity's GUI system and the SECTR AUDIO window may become sluggish. If this happens, simply collapse the any folders you're not working in and, if you can, turn off full details mode.

Hierarchy Panel

The hierarchy panel shows all of the AudioBuses and AudioCues in the project. The tree structure shows you where individual assets fall in that hierarchy. Left clicking on any Cue or bus will show you its information in the Properties Panel (see below).

Right clicking anywhere in the panel gives you options for modifying, creating, and deleting the assets displayed here.

Properties Panel

As its name suggests, the properties panel displays all of the information relevant to the object selected in the Hierarchy Panel.

When an AudioCue is selected, a version of its inspector is drawn in the Properties Panel. You can edit things here exactly as you would in the regular inspector, with full support for Undo and the like.

When an AudioBus is selected, the Properties Panel becomes a mixing view, displaying sliders for all of the AudioBuses in the project. If this is too many for you, you can use the search dialog at the top to filter the list down to what you are interested in.

Keyboard Shortcuts

The SECTR Audio Window also supports keyboard shortcuts for common functions. These shortcuts are:

- Arrow Keys: Navigate hierarchy and clip views.
- Enter/Return: Rename selected hierarchy item.
- Ctrl+D: Duplicate selected hierarchy item.

Creating Audio

While the SECTR AUDIO window makes it easy to create AudioCues, AudioCues themselves have a number of properties that you may want to understand before you get too far. This section describes those properties in detail, but you can also refer to the in-editor tool tips if you forget anything here.

Basic Properties

- **Template:** If set, this is another AudioCue that this AudioCue will take all of its properties from, except for the AudioClips and the Bus. Templates are a powerful way to manage a lot of sounds with just a few files.
- **Loops:** If this clip should loop on playback. Loops affects the culling behavior, as described below.
- **HDR:** Denotes if this AudioCue should be mixed in high dynamic range or low dynamic range, as described in the Mixing section.
- **Volume and Loudness:** LDR sounds will have Volume, while HDR sounds have Loudness. In both cases, you can specify a min and max Volume/Loudness. SECTR AUDIO will choose a unique value from this range on every playback.
- **Pitch:** Like loudness, this allows you to supply a min and match pitch offset. This can be a very cheap way to create additional variation.
- **Fade In Time** and **Fade Out Time:** Specifies how long to fade in when played, and how long to fade out when stopped.
- **Spatialization:** Determines how this sound is positioned in the surround field.
    - Simple2D sounds are just that, 2D sounds with no position at all.
    - Linear3D sounds are your standard 3D sounds.
    - Occludable3D sounds are like Linear3D, but they also can be Occluded.
    - Infinite3D sounds have 3D position, but do not attenuate at all.

Advanced Properties

- **Delay:** Number of seconds to wait after Play() to actually start the audio.
- **Play Probability:** A chance that the cue will actually create a sound when Play() is called. Helps thin out the mix when set to be < 100%.
- **Spread:** Determines the range of speakers from which the sound is audible.
- **Max Instances:** The maximum number of instances of this Audio Cue that can exist at any one time.
- **Priority:** Internal FMOD priority, as described in the Unity docs.
- **Bypass Effects:** When true, Cue will ignore reverb zones or filters. (Pro Only)
- **Prefab:** The prefab that defines filters and other custom behavior for this Cue.

2D Properties

- **Pan2D:** Moves the sound around the speaker field.

3D Properties

- **Falloff**: Determines how this sound attenuates with distance. The options are linear and logarithmic, which behave the same as Unity AudioSource.
- **Min Distance:** The distance at which sounds start attenuating.
- **Max Distance:** The distance at which sounds are no longer audible.
- **Occlusion Scale:** The amount by which this Cue is affected by the global occlusion settings.
- **Doppler Level**: Scales the amount of doppler applied to this sound.
- **Proximity Limit**: The maximum number of instances that can be within Proximity Range of one another. Does nothing if set to 0.
- **Proximity Range**: The minimum distance between instances, if Proximity Limit is set to be greater than 0. See Mixing below for more information.

Audio Clips

- **Clip List:** Set of AudioClips to select from when playing this Cue.
- **Clip Volume:** Individual volume for each AudioClip. Allows for per-Clip mixing.
- **HDR Curve:** Displays the loudness envelope for Clips in HDR cues. Can be user edited for finer grained control over the HDR mix.
- **Playback Mode:** Determines the rules for how AudioClips are selected. The default behavior is random, but shuffle, loop, and ping pong are also options.

Playing Sounds

Once you've created some audio assets, it's time to start playing them. Before you can play sounds in SECTR AUDIO, you need to set up your scene.

Preparing for Playback

To play sounds in SECTR AUDIO, all you need to do is add a SECTR AUDIOSystem component to your Listener. Once you've created an AudioSystem component, hook up an Audio Bus to the Master Bus attribute.. And that's it!

If you want to use PropagationSource or try out graph based (i.e. non-raycast) occlusion, you'll need to create some Sectors and Portals, too. For more information on how to do so, see the SECTR CORE documentation.

Audio Sources

SECTR AUDIO includes a number of audio components designed to handle audio playback in common cases. These SECTR AUDIOSources are similar to Unity AudioSources, except they support a much wider range of features than just point source playback. The following explains what each one does:

- **Point Source:** Plays a sound at a 3D point. Very similar to the Unity Audio Source.
- **Region Source:** Plays a sound at a point on a collider nearest to the Listener. Great way to represent volumetric sounds.
- **Spline Source:** Plays a sound at a point on a 3D spline nearest to the Listener. Very useful for rivers and roads and the like.
- **Propagation Source:** This source models how audio bounces around through an environment. Sounds amazing, but requires Sectors and Portals to work.
- **Trigger Source:** Plays a sound when a player enters a Trigger. Good way to do simple sound scripting.
- **Impact Source:** Plays sounds when a rigid body collides with the world. Great way to tie sounds into the physics.

Audio Environments

Although Audio Sources are great for sounds that have a specific 3D position, games often want to have a base layer of "2D" or non-specific sound playing (sometimes called an ambience). To achieve this effect, SECTR AUDIO provides a set of AudioEnvironment objects that make creating ambient audio easy.

Each Audio Environment contains the settings for an AudioAmbience. An AudioAmbience simply contains a background loop, a list of one shots to play randomly, and a range of time between one shots. In general, the background loop should be Simple2D and the one shots should be Infinite 3D.

During the game, there may be multiple Audio Environments active in the world, but only one of their AudioAmbiences will be audible at a time. To achieve this, the AudioSystem has a stack of AudioAmbiences, the top of which is audible. When a new AudioEnvironment becomes active, its AudioAmbience is placed on top of the stack. When that AudioEnvironment becomes inactive, it removes its AudioAmbience from the stack, wherever it is. This approach allows AudioEnvironments to be overlapped, nested, etc and still sound correct.

Gameplay Events

For sounds triggered by gameplay events, like weapon impacts or UI sounds, programmers should call SECTR_AudioSystem.Play() directly, as this is the most efficient way to play sounds in SECTR_Audio. If this sound loops, or needs to be updated after it is played, the AudioSystem will return an AudioCueInstance that you can use to access the instance of this particular AudioCue.

Music

Music is an important component to any game audio, but SECTR AUDIO takes a simple approach to its music system. Music, in SECTR AUDIO is simply a special way to play an AudioCue, and the AudioSystem guarantees that no more than one will play at once. While simple, this system is very effective for the vast majority of ingame music.

Mixing

Mixing is one of the most important, but generally time consuming aspects of creating interactive audio. SECTR AUDIO includes several tools to both make for higher quality, and less time consuming mixing.

Bus Hierarchy

SECTR AUDIO includes a standardize hierarchical mixer that should be familiar to most sound designers. Each AudioCue in the game has a parent bus, and its final volume is determined by the volume of its parent and all of its parents' parents. This hierarchy makes it easy to make both very broad and very specific mixing changes easily.

Instance Limits

SECTR AUDIO also allows you to control the number of instances of an AudioCue, both globally and, for 3D sounds, within a region of space. Used together, these instance limits can help "thin out" the mix and prevent it from becoming cluttered.

The first limit is called Max Instances. Max Instances determines the maximum number of times an AudioCue can be playing concurrently. Once this limit is reached, future requests will fail until one of the current instances stops playing.

The second limit is the Proximity Limit. This feature is for 3D sounds only, and, combined with Proximity Range, allows you to specify the number of instances of an Audio Cue that can play within a certain distance from one another. For example, a Proximity Limit of 2 and a Proximity Range of 1, means that no more than 2 instances of a sound can play within 1 meter of each other. If a third sound plays, it it would be 1 meter or less from two current instances, it would simply fail to play.

HDR Audio

SECTR AUDIO also includes an optional, high dynamic range (HDR) mixer. Individual Cues can be tagged as HDR. When tagged this way, they are given a physically accurate representation of volume – called loudness. Unlike volume, which goes

from 0 to 1, loudness is a logarithmic representation of the sound pressure level, called dB(SPL). Loudnesses can be set to any value that sounds good, but it's good to start with real world values, some examples of which can be found on Wikipedia.

While the Loudness property of the Cue establishes a baseline for the Loudness, real sounds are rarely uniformly loud. Real world sounds often have a mix of louder and quieter sections. To represent this variation, SECTR AUDIO can compute HDR Keys, which are basically a compressed representation of each AudioClip's volume. These HDR keys are not required to play HDR audio, but they will make the mix sound much better. HDR keys can be baked in the Cue Inspector and in the Audio Window. If HDR keys are not baked, SECTR AUDIO will let you know.

When HDR sounds are enabled, they still have to be converted to LDR sounds before playback. To do this conversion, SECTR AUDIO uses a sliding "window", which functions like the exposure of a camera. The top of this window is the loudness of the loudest sound currently playing. The bottom of the window is determined by the HDR Window Size attribute in the Audio System. The final volume of the sound is determined by where its loudness falls within this window. That means that a particular AudioCue will sound louder or softer based not just on its own loudness but on the loudness of all of the other active sounds.

The dynamic nature of HDR audio creates a very realistic, impressive effect, but it takes time to learn if you're familiar with traditional, LDR mixing. If you try out HDR audio, make sure to play around with it for a while to get the hang of how this kind of mixing works. There are also a number of good articles on the internet about how to think about HDR audio mixing.

Spatial Effects

Interactive audio always takes place within a 3D (or 2D) context. The geometry of this environment influences how the sounds in it are perceived. Stock Unity has a few tools to help with this, specifically Audio Reverb and distance based Low Pass (for Pro users). SECTR AUDIO adds several more powerful tools.

Near 2D Blend

The simplest spatial effect in SECTR AUDIO is the ability to blend 3D sounds that are close to the AudioListener into 2D sounds, called Near 2D Blend. In the real world, we almost always hear sounds in both ears, especially for sounds close by. The ear opposite the sound source hears sound waves as they reflect off nearby objects and as they pass through the skull. This effect is called a head related transfer function (HRTF). Doing a true HRTF is complicated, but SECTR AUDIO's Near 2D Blend is a very simple approximation. One of the benefits of this feature is that sounds that pass by the camera no longer make a harsh pop as they go from one side of the stereo field to the other.

Occlusion

In the real world, there are often objects in between things that emit sounds and the people that hear them. These obstructions "occlude" the source, and change the character of the sound, generally making the sound quieter while removing high frequency components of the sound.

To enable audio occlusion in SECTR AUDIO, two things must be done. First, any AudioCue that should be occludable needs to have its Spatialization set to Occludable3D. Occlusion requires additional CPU so it must be enabled individually. Second, an occlusion mode must be set in the AudioSystem.

SECTR AUDIO has two mechanisms for computing audio occlusion. You can chose which one (or both) to be active at once. When ray casting is enabled, the AudioSystem will periodically do ray checks against the scene collision from each Occludable Source to itself. When Graph occlusion is enabled, the AudioSystem will use the Sector/Portal graph to determine if the sound is audible. This solution will

take things like the closed flags on doors into account.

In both cases, the time between tests is determined by the RetestInterval property of the AudioSystem. Making the RetestInterval shorter will increase the accuracy of the occlusion calculations, but will increase the CPU cost for all Occludable3D AudioCues.

Propagation

In complex interior environments, the sounds we hear may not even come directly from the source. Imagine standing in a room at the end of an L shaped hallway. At the other end of the L shaped hallway is another room, in which a stereo is blasting music. In most games, you would hear the music coming directly from the point of the stereo, but in the real world the sound would appear to be coming through the doorway that leads into the hall, because the sound of the reflections down the hall are louder than the sound waves that have to travel through walls. This phenomena of how reflected sound moves through an environment is called audio propagation.

SECTR AUDIO has support for audio propagation thanks to the PropagationSource component. You can place one of these anywhere in your level, provided you've set up Sectors and Portals, and the sounds will appear to reflect properly through your rooms and hallways.

SECTR uses the Sector/Portal graph to create a fast approximation of real sound propagation. Even this approximation is more expensive than playing a simple 3D sound, so PropagationSources should be used sparingly, only where they make a discernable impact to the player.

Lastly, PropagationSource performs its own occlusion calculations, so you should only use Linear3D, not Occludable3D AudioCues.

Custom Filters And Effects

Unity Pro includes Audio Filters, which allow developers to add realtime DSP effects like reverb and echo to their game audio. SECTR allows you to apply these filters (and other effects) to AudioCues using the Prefab attribute.

The Prefab attribute allows you to tell SECTR to use a custom Prefab to configure the instances of an AudioCue, rather than the default behavior which uses very simple GameObjects (basically just a Transform and AudioSource). You can find the Prefab attribute in the the Advanced section of the Cue properties.

Note that while the Prefab attribute is most useful as a way to add AudioFilters to an AudioCue, you can include whatever components you like. As an example, you could add a Particle System component to the Prefab for certain AudioCues to ensure that they play a visual effect every time the sound is triggered.

Realtime Parameter Control

RPC is an advanced audio technique intended for cases that require AudioCues to change directly based on game state. RPC allows the game programmers to pass in parameter values (simple floats) which the audio designer can then use to modify properties of the AudioCue and any other components that come from the Prefab (if specified, see Custom Filters and Effect chapter).

To add RPC to an AudioCue, simply go to the Control Parameters section and press the Add Parameter button. Next, enter the name of the parameter and choose what attribute you would like to effect. You can give the Parameter any name that you like, but there are two special names which SECTR will set for you. The first is distance, which is the number of units between listener and AudioCue instance, and the second is time, which is the number of seconds that have elapsed since the AudioCue started playing.

If you want to modify the property from a Prefab, choose Attribute. Attribute will automatically activate a new GUI which automatically detects the RPC compatible attributes on your custom prefab. If you see an attribute that you think should support RPC but does not, please email support@makecodenow.com.

Be sure to set a good Default value for the parameter. This is the value that will be set when the AudioCue is first played, and will be the parameter value until a programmer specifies a new value.

Next, it's time to modify the actual parameter curve. To edit the curve, simply double click the curve in the inspector or Audio Window. In SECTR AUDIO, parameters multiply the base value of the attribute, which means that 1 means "do nothing", a value of 2 means "double the base value" and a value of 0.5 means "halve the base value".

The final step in using RPC is for your programmer to set a value for the parameter on a per-instance basis, which they can do simply by calling the SECTR_AudioInstance.SetParameter() function. Note that parameter names are case sensitive.

Culling and Optimization

In order to remain efficient, SECTR AUDIO avoids doing work whenever possible, which mostly boils down to effective culling and virtualization.

One Shot Culling

One shot sounds are generally fairly short, and it's likely that if the sound was inaudible when played, then it will be inaudible when it finishes, too. As such, SECTR AUDIO will pre-cull any non-looping sound if the sound is played beyond its Max Distance, or if it is an HDR sound but is too quiet to be audible.

Loop Virtualization

Looping sounds last forever (or until stopped) so SECTR AUDIO always produces AudioCueInstances for looping sounds. However, SECTR AUDIO will virtualize these sounds when they become inaudible. Virtual instances appear identical to the rest of the game code, but the require the bare minimum of resources to maintain.

The AudioSystem will periodically test to see if a looping sound should become virtual or become real. The amount of time between tests is determined by the RetestTime property in the AudioSystem. Lower numbers mean less latency and more aural accuracy, but it also consumes more CPU time. SECTR AUDIO does try to minimize the impact of these tests by generating random times within the range specified by RetestTime.

Programmer's Guide

For programmers who plan to integrate their gameplay and systems code with SECTR AUDIO, this chapter explains the best practices for using the included API.

The most important thing to know as a programmer is that your code should talk directly to the AudioSystem whenever possible. If you want to play a simple sound, call SECTR_AudioSystem.Play(), don't create a new GameObject and add a PointSource component to it. Using the AudioSystem API directly allows you to play sounds with the minimum possible overhead. The one exception to this rule is if playing propagation sources. If you want to play propagation sources, you should use a Propagation Source component and call its Play() and Stop() functions.

When you play sounds directly through the AudioSystem, you will be given an AudioCueInstance, which acts like a handle to the currently playing sound. You are not required to use this handle, but if you want to modify the sound after playback, you will need to hold on to the instance handle. Note that because SECTR AUDIO culls sounds that are guaranteed to be inaudible, you may receive a null AudioCueInstance even when playing a valid AudioCue.

Because AudioCueInstances are handles, they may stop playing and/or become null at any time. Always use the bool operator (i.e. if(someInstance) …) before modifying an instance. Instance handles are pooled and recycled, so be sure to clear your handles when you are done with them (if the provided API does not do it for you). In particular, handles to one-shot sounds can be dangerous to hold on to in the current implementation of the handle system.

If you want to play sounds when a level is beginning, put your code in Start, not in OnEnable. The AudioSystem initializes itself in OnEnable and there is no guarantee that your component will be enabled before or after the AudioSystem.