API REFERENCE MANUAL

Version 1.2.0

Chapter 1

Module Documentation

1.1 CORE

The foundation of the SECTR suite of Unity extensions.

Classes

- class SECTR_Door

  Implements a basic door component that is Portal aware. Also, provides an interface that more complex doors can implement.
- class SECTR_Geometry

  A library of useful geometric functions.
- class SECTR_Graph

  A set of static utility functions used to traverse the Sector/Portal graph.
- class SECTR_Hull

  Abstract base class that implements planar, convex hulls, for use in SECTR_Portal, SECTR_Occluder and other client classes.
- class SECTR_Member

  Member represents anything that can be part of a SECTR_Sector, including Sectors themselves.
- class SECTR_Portal

  Portals define the logical and geometric connection between two SECTR_Sector objects.
- class SECTR_PriorityQueue< T >

  Implements a priority queue in terms of a binary heap.
- class SECTR_Sector

  Sectors represent discrete sections of the world, connected to one another by SECTR_Portal objects.

Detailed Description

The foundation of the SECTR suite of Unity extensions. It includes all of the tools necessary to quickly and easily add Sectors and Portals to your Unity-based game, as well as full source code that you can build your own unique features.

1.2 VIS

A high performance, low memory, occlusion culling solution for Unity.

Classes

- class SECTR_Culler

  Vestigial component from older version of SECTR. Left intact only for backwards compatability.
- class SECTR_CullingCamera

  CullingCamera is the workhorse of SECTR Vis, culling objects by propagating Camera data down through the Sector/-Portal graph and into individual SECTR_Culler objects.
- class SECTR_LOD

  Implements a simple Level of Detail (LOD) system for SECTR objects.
- class SECTR_Occluder

  An Occluder represents a visual obstruction. It will hide any objects behind it (from the perspective of the current SECTR_Culler).

Detailed Description

A high performance, low memory, occlusion culling solution for Unity. SECTR Vis supports the complete set of Unity rendering primitives including lights, shadows, particles, meshes, and terrain. SECTR Vis is fully dynamic, requires no tedious baking process, and works with both Unity Free and Pro. Heavily optimized, SECTR Vis is a great solution for all platforms, especially tablets, smart phones, portables, and "last gen" consoles.

1.3 STREAM

Makes it easy to save memory, increase performance, and decrease load times by splitting your scene into multiple chunks and streaming them in and out in realtime.

Classes

- class SECTR_StreamExport

  A set of static utility functions for exporting scenes and doing other stream related processing.
- class SECTR_Chunk

  Chunk is the loadable/streamable version of a SECTR_Sector. The Chunk manages loading and unloading that data, usually at the request of a Loader component.
- class SECTR_ChunkRef

  Allows for faster finding of loaded chunks.
- class SECTR_GroupLoader

  Allows users to group a set of Sectors, loading and unloading them as if they were a single Sector.
- class SECTR_Hibernator

  Automatically enables and disables components on itself when the SECTR_Sector it's part of are (un)loaded.
- class SECTR_LightmapRef

  Stores the references to lightmap textures in an exported Chunk.
- class SECTR_Loader

  Provides an abstract base class for classes that load data from SECTR_Chunk components.
- class SECTR_LoadingDoor

  Extends the basic SECTR_Door with awareness of streaming SECTR_Chunks. This door won't open unless the SECTR_Chunks on both sides of the door's SECTR_Portal are loaded.
- class SECTR_NeighborLoader

  Loads SECTR_Chunk components that are in the current or adjacent SECTR_Sector.
- class SECTR_RegionLoader

  (Un)loads Chunks within a given volume. Can be set to optionally not touch Sectors that are not part of the terrain grid.
- class SECTR_StartLoader

  Loads SECTR_Chunk components that this object is in at Start and nothing more.
- class SECTR_TriggerLoader

  (Un)loads a list of SECTR_Chunk objects based on Unity Trigger events.

Detailed Description

Makes it easy to save memory, increase performance, and decrease load times by splitting your scene into multiple chunks and streaming them in and out in realtime. SCTR Stream is ideal for titles targeting memory limited devices like tablets and smartphones, or games on any platform that want to have large, seamless worlds free of loading screens. SCTR Stream includes all of the editor tools and runtime components needed to split your scenes up and stream them at runtime. Games that already have already setup Sectors and Portals with SCTR_Core can be streaming in seconds. SECTR Stream even handles streaming "global" objects like lightmaps, nav meshes, and light probes.

1.4 AUDIO

Brings the latest, cutting edge audio production tools and technologies to Unity.

Classes

- class SECTR_AudioBus

  Represents the configuration of a particular mixing bus, which can be used to bulk mix SECTR_AudioCue instances.
- class SECTR_AudioCue

  A Cue is the atomic, playable object in SECTR Audio. It encapsulates all of the data necessary for randomization, spatialization, mixing, etc.
- struct SECTR_AudioCueInstance

  A handle to and interface for instances of SECTR_AudioCue.
- class SECTR_AudioEnvironment

  An abstract base class for spatial components that add and remove SECTR_AudioAmbience objects from the main SECTR_AudioSystem.
- class SECTR_AudioEnvironmentTrigger

  Activates a SECTR_AudioAmbience whenever the sibling trigger volume is entered.
- class SECTR_AudioEnvironmentZone

  Activates a SECTR_AudioAmbience whenever a player enters an AudioReverbZone.
- class SECTR_AudioSource

  An abstract base class for all components in SECTR Audio can be placed within the scene. AudioSource also provides a common interface to the user, and basic functions like play, stop, etc.
- class SECTR_AudioSystem

  The beating heart of SECTR_Audio, SECTR_AudioSystem provides all of the services necessary to play sounds and music, control the mix, etc.
- class SECTR_CharacterAudio

  Plays audio based on character events.
- class SECTR_ComputeRMS

  Internal class to compute per-second RMS values of sounds and store them in HDR keys.
- class SECTR_DoorAudio

  Extends the basic SECTR_Door with sounds that play on state transitions.
- class SECTR_ImpactAudio

  Playes a SECTR_AudioCue when a physics impact is detected.
- class SECTR_MusicTrigger

  Makes the specified music active when a trigger is entered.
- class SECTR_PointSource

  Plays a SECTR_AudioCue at this point in the world.
- class SECTR_PropagationSource

  Propagation Source simulates the complex phenomena of audio reflections in a closed space.
- class SECTR_RegionSource

  Playes a SECTR_AudioCue within a 3D volume.
- class SECTR_SplineSource

  Plays the specified SECTR_AudioCue at the nearest point along a spline to the listener.
- class SECTR_TriggerSource

  Playes a SECTR_AudioCue when a trigger is activated.

Detailed Description

Brings the latest, cutting edge audio production tools and technologies to Unity. SECTR Audio includes 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. 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.

1.5 DEMO

Components used in the SECTR Demos.

Classes

- class SECTR_CharacterMotor

  C# adaptation of the Unity sample CharacterMotor, with custom tweaks and extensions.
- class SECTR_DemoUI

  A simple harness for demo messages and input handling.
- class SECTR_FogDisable

  Disables fog in the associated camera (for use in the PiP camera).
- class SECTR_FPController

  Simple abstract base class for first person style controllers.
- class SECTR_FPSController

  A simple FPS style character controller.
- class SECTR_GhostController

  Implements a standard spectator/fly camera.
- class SECTR_Wanderer

  A component that will wander the scene by pathing through the Sector/Portal graph.

Detailed Description

Components used in the SECTR Demos. These simple gameplay components are designed for use in the product demos, but may be useful for anyone as basic gameplay primitives.

Chapter 2

Class Documentation

2.1 SECTR_Member.Child Class Reference

Simple data structure to represent the important information about one of the children of a SECTR_Member.

Public Attributes

- GameObject gameObject

  The game object this member was created from.
- int gameObjectHash

  Hash of GameObject ID.
- SECTR_Member member

  The Member this child belongs to.
- Renderer renderer

  Renderer component of Member child. Can be null.
- int renderHash

  Hash of render component ID.
- Light light

  Light component of Member child. Can be null.
- int lightHash

  Hash of light component ID.
- Terrain terrain

  Terrain component of Member child. Can be null.
- int terrainHash

  Hash of terrain component ID.
- Bounds rendererBounds

  Cached world space bounds of the renderer component.
- Bounds lightBounds

  Cached world space bounds of the light component.
- Bounds terrainBounds

  Cached world space bounds of the terrain component.
- bool shadowLight

  Cached value of ability to create dynamic shadows.
- bool rendererCastsShadows

  Cached value renderer dynamic shadow casting.
- bool terrainCastsShadows

  Cached value of terrain ability to cast dynamic shadows.

Detailed Description

Simple data structure to represent the important information about one of the children of a SECTR_Member.

2.2 SECTR_Graph.Node Class Reference

Represents a Node in the Sector/Portal graph. Contains useful data for implementing traversals.

Inherits IComparable< Node >.

Public Member Functions

- int CompareTo (Node other)

  Comparison function for two Nodes. Used in A.*

  Parameters

other	*The Node against to compare ourselves.*

  Returns

  The relative ordering of this and another Node.

Static Public Member Functions

- static void ReconstructPath (List< Node > path, Node currentNode)

  Utility function for reconstructing a path from a set of nodes.

  Parameters

path	*The List to populate with the path.*
currentNode	*The Node from which to start the path generation.*

Detailed Description

Represents a Node in the Sector/Portal graph. Contains useful data for implementing traversals.

2.3 SECTR_AudioAmbience Class Reference

Defines the data specific to a particular SECTR_AudioEnvironment.

Public Attributes

- SECTR_AudioCue BackgroundLoop = null

  The looping 2D cue to play as long as this ambience is active.
- List< SECTR_AudioCue > OneShots = new List<SECTR_AudioCue>()

  A list of one-shots that will play randomly around the listener.
- Vector2 OneShotInterval = new Vector2(30f, 60f)

  The min and max time between one-shot playback.
- float Volume = 1f

  The a volume scalar for the Cues in this Ambience. Combines with the base Cue volume.
- bool UseOneShotCuesProbability = true

  When enabled the higher probability one shots will be more likely to play, gets the probability from the cue.

Detailed Description

Defines the data specific to a particular SECTR_AudioEnvironment.

The goal of environmental audio (also known as ambient audio) is to create a base layer of ambient sound effects, effects which always play even if there is not much going on in the game.

In SECTR, AudioAmbiences contain two sets of cues that can help create this baseline. Each AudioAmbience can have a background loop, which will be played as a looping, 2D sound as long as that AudioAmbiences is the current, highest priority AudioAmbiences in the scene. The AudioAmbiences can also have one or more one-shot sounds that will fire randomly. If these one-shots are marked as Infinite3D, then they will be randomly position in the surround field, giving the impression of sounds playing all around the player.

2.4 SECTR_AudioBus Class Reference

Represents the configuration of a particular mixing bus, which can be used to bulk mix SECTR_AudioCue instances.

Inherits ScriptableObject.

Public Member Functions

- bool IsAncestorOf (SECTR_AudioBus bus)

  Determines whether this instance is an ancestor of the specified bus.

  Parameters

bus	*The bus to check ancestry of.*

  Returns

  Returns true if this bus is an ancestor of the specified bus.
- bool IsDecendentOf (SECTR_AudioBus bus)

  Determines whether this instance is a decendent of the specified bus.

  Parameters

bus	*The bus to check ancestry of.*

  Returns

  Returns true if this bus is a decendent of the specified bus.
- void ResetUserVolume ()

  Resets the User Volume to 1 for this bus and all it's children.

Public Attributes

- float Volume = 1

  The volume of this bus, between 0 and 1.
- float Pitch = 1

  The pitch of this bus, between 0 and 2.

Properties

- float UserVolume `[get, set]`

  Accessor for the user volume. This is a volume that is not saved and is applied on top of the volume set in the original resource.
- bool Muted `[get, set]`

  (Un)Mutes this bus, and all of the sounds in it.
- float EffectiveVolume `[get, set]`

An optimization that returns the current, flattened bus volume.

- float EffectivePitch `[get, set]`

  An optimization that returns the current, flattened bus pitch.

- SECTR_AudioBus Parent `[get, set]`

  Accessor for this Bus's parent (if any).

- List< SECTR_AudioBus > Children `[get]`

  Returns the list of buses that are children of this bus.

Detailed Description

Represents the configuration of a particular mixing bus, which can be used to bulk mix SECTR_AudioCue instances.

Mixing buses are stored in a hierarchy, where the settings cascade down the hierarchy (i.e. if a bus is muted, then so are all of its children). This hierarchical relationship makes it easier to mix the game than if the volumes of every cue needed to be adjusted individually. Buses can also be used at runtime to do things as simple as providing user controlled FX/Music/Voice sliders, to completely dynamic mixing.

2.5 SECTR_AudioCue Class Reference

A Cue is the atomic, playable object in SECTR Audio. It encapsulates all of the data necessary for randomization, spatialization, mixing, etc.

Inherits ScriptableObject.

Public Types

- enum PlaybackModes { PlaybackModes.Random, PlaybackModes.Shuffle, PlaybackModes.Loop, PlaybackModes.PingPong }

  Types of rules for picking the next AudioClip.

- enum FalloffTypes { FalloffTypes.Linear, FalloffTypes.Logrithmic }

  Types of rules for picking the next AudioClip.

- enum Spatializations { Spatializations.Simple2D, Spatializations.Infinite3D, Spatializations.Local3D, Spatializations.Occludable3D }

  Ways to spatialize the sound (i.e. position in the surround field)

Public Member Functions

- ClipData GetNextClip ()

  Returns the next AudioClip to be played, as determined by the PlaybackMode.

Public Attributes

- List< ClipData > AudioClips = new List<ClipData>()

  List of Audio Clips for this Cue to choose from.

- PlaybackModes PlaybackMode = PlaybackModes.Random

  The rules for selecting which audio clip to play next.

- bool HDR = false

  Determines if the sound should be mixed in HDR or LDR.

- Vector2 Loudness = new Vector2(50f, 50f)

  The loudness, in dB(SPL), of this HDR Cue.

- Vector2 Volume = Vector2.one

The volume of this Cue.

- Vector2 Pitch = Vector2.one

  The pitch adjustment of this Cue.

- bool Loops = false

  Set to true to auto-loop this Cue.

- int Priority = 128

  Cue priority, lower is more important.

- bool BypassEffects = false

  Prevent this Cue from recieving Audio Effects.

- int MaxInstances = 10

  Maximum number of instances of this Cue that can be played at once.

- float FadeInTime = 0f

  Number of seconds over which to fade in the Cue when played.

- float FadeOutTime = 0f

  Number of seconds over which to fade out the Cue when stopped.

- Spatializations Spatialization = Spatializations.Local3D

  Sets rules for how to spatialize this sound.

- float Spread = 0

  Expands or narrows the range of speakers out of which this Cue plays.

- float Pan2D = 0

  Moves the sound around the speaker field.

- FalloffTypes Falloff = FalloffTypes.Linear

  Attenuation style of this clip.

- float MaxDistance = 100

  The range at which the sound is no longer audible.

- float MinDistance = 10

  The range within which the sound will be at peak volume/loudness.

- float DopplerLevel = 0

  Scales the amount of doppler effect applied to this Cue.

- int ProximityLimit = 0

  Prevents too many instances of a cue playing near one another.

- float ProximityRange = 10

  The size of the proximity limit check.

- float OcclusionScale = 1f

  Allows you to scale down the amount of occlusion applied to this Cue (when occluded).

- float PlayProbability = 1f

  The chance that this cue will actually make a sound when played.

- Vector2 Delay = Vector2.zero

  Random delay before start of playback.

- GameObject Prefab

  Special prefab to use when playing this Cue. Useful for adding effects to this source.

- List< SECTR_CueParam > ControlParams = new List<SECTR_CueParam>()

  Control parameters for this Cue.

Properties

- SECTR_AudioCue Template [get, set]

  Accessor for the Template cue of this Cue. If set, the Template will override all properties of the Cue except for the list of AudioClips and the parent Bus.
- SECTR_AudioBus Bus [get, set]

  Accessor for the Bus of this Cue.
- SECTR_AudioCue SourceCue [get]

  Returns the Cue that determines the 2D and 3D properties, will always be this Cue or its Template.
- bool Is3D [get]

  Returns true if this Cue is Local3D or Infinite3D.
- bool IsLocal [get]

  Returns true if this Cue is Simple2D or Infinite3D.
- int ClipIndex [get]

  Returns the index of the currently playing AudioClip.

Detailed Description

A Cue is the atomic, playable object in SECTR Audio. It encapsulates all of the data necessary for randomization, spatialization, mixing, etc.

Cue has a number of properties, but they fall into three basic categories: properties common to all cues (like pitch and spatialization), spatial properties (i.e. 2D or 3D specific attributes), and properties related to the management and playback of AudioClips. These categories are visible in the code below, and in the custom inspector in the Unity Editor.

Because games often have many sounds with the same properties but different AudioClips, AudioCue provides a simple templating system. Templates are somewhat like a simple, audio specific version of Unity prefabs, though they do not allow per-attribute overrides. Because of this feature, programmers who need to access the properties of a given SoundCue should be careful to use the SourceCue property as that will always return the AudioCue that whose properties will be used by the SECTR_AudioSystem.

Member Enumeration Documentation

enum SECTR_AudioCue.FalloffTypes

Types of rules for picking the next AudioClip.

Enumerator

Linear Audio attenuates linearly between Min and Max distances.

Logrithmic Audio attenuates logrithmically between Min and Max distances.

enum SECTR_AudioCue.PlaybackModes

Types of rules for picking the next AudioClip.

Enumerator

Random Select an AudioClip at random.

Shuffle Select an AudioClip at random, but do not repeat any until all have played.

Loop Play AudioClips in order, starting over at the beginning when all are played.

PingPong Play AudioClips in ascending and then descending order.

enum SECTR_AudioCue.Spatializations

Ways to spatialize the sound (i.e. position in the surround field)

Enumerator

Simple2D The most basic 2D sound. Not affected by 3D position at all.

Infinite3D A 3D sound with direction by no distance attenuation. Ideal for random ambient one shots.

Local3D The most basic 3D sound. Spatialized and attenuated by 3D position.

Occludable3D Same behavior as a 3D sound, but may be affected by Occlusion calculations.

2.6 SECTR_AudioCueInstance Struct Reference

A handle to and interface for instances of SECTR_AudioCue.

Inherits SECTR_IAudioInstance.

Public Member Functions

- void Stop (bool stopImmediately)

  Ends playback of the specified audio instance.
  Parameters

stopImmediately	*Immediate stop will ignore any Fade Out set on the original SECTR_AudioCue.*

- void ForceInfinite ()

  Forces a 3D sound to act as infinite 3D. For very special case uses.
- void ForceOcclusion (bool occluded)

  Forces occlusion on or off. For very special use cases.
- void SetParameter (string param, float value)

  Sets the value of a control parameter.
- SECTR_IAudioInstance GetInternalInstance ()

  Returns internal instance. For very special use cases.

Properties

- bool Active `[get]`

  Does this instance refer to an active, valid sound, or is it a dead handle.
- Vector3 Position `[get, set]`

  Accessor for the instance's world space position.
- Vector3 LocalPosition `[get, set]`

  Accessor for the instance's local space position.
- float Volume `[get, set]`

  Accessor for the volume of the instance. This volume will be combined with other volumes (like from the Bus hierarchy) to produce the final volume.
- float Pitch `[get, set]`

  Accessor for the pitch of the instance. This pitch will be combined with the pitch from the Bus hierarchy to produce the final pitch.
- bool Mute `[get, set]`

  Accessor for the mute state of the instance. Mute state will be combined with the bus hierarchy mute state to produce a final volume.
- bool Pause `[get, set]`

Accessor for the pause state of the instance. Pause state will be combined with the system pause state to produce a final value.

- float TimeSeconds `[get, set]`

  Accessor for the elapsed playback time in seconds.

- int TimeSamples `[get, set]`

  Accessor for the elapsed playback time in samples.

Detailed Description

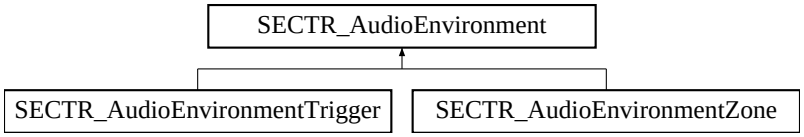
A handle to and interface for instances of SECTR_AudioCue.

A unique SECTR_AudioCueInstance is returned each time a SECTR_AudioCue is played. This instance serves as a handle with which to manipulate the instance after initial playback (if so desired). Client systems are free to ignore this return value, in which case the sound is assumed to be "fire-and-forget". Looping sounds, however, will not auto-stop themselves (until the end of the game) so programmers should take care to stop handle them properly.

2.7 SECTR_AudioEnvironment Class Reference

An abstract base class for spatial components that add and remove SECTR_AudioAmbience objects from the main SECTR_AudioSystem.

Inheritance diagram for SECTR_AudioEnvironment:



Public Attributes

- SECTR_AudioAmbience Ambience = new SECTR_AudioAmbience()

  The configuraiton of the ambient audio in this Reverb Zone.

Properties

- bool Active `[get]`

  Returns true if this AudioEnvironment has put its Ambience on the stack.

Detailed Description

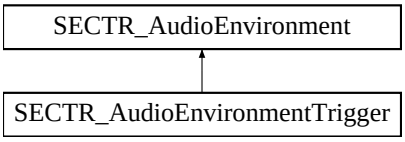
An abstract base class for spatial components that add and remove SECTR_AudioAmbience objects from the main SECTR_AudioSystem.

AudioEnvironments interact directly with the AudioSystem's stack of active Ambiences. When the Audio Environment is activated, its AudioAmbience is pushed onto the SECTR_AudioSystem's stack of active Audio Environments, but when the player leaves, the Audio Environment is removed from the stack, wherever it is. This allows Audio Environments to overlap and even be nested within one another.

2.8 SECTR_AudioEnvironmentTrigger Class Reference

Activates a SECTR_AudioAmbience whenever the sibling trigger volume is entered.

Inheritance diagram for SECTR_AudioEnvironmentTrigger:



Public Attributes

- SECTR_AudioAmbience Ambience = new SECTR_AudioAmbience()

  The configuraiton of the ambient audio in this Reverb Zone.

Properties

- bool Active `[get]`

  Returns true if this AudioEnvironment has put its Ambience on the stack.

Detailed Description

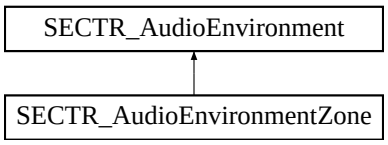
Activates a SECTR_AudioAmbience whenever the sibling trigger volume is entered.

AudioEnvironmentTriggers activate based on the standard Unity trigger events. As such, they will work with any shaped collider, provided it's marked as a trigger. As with all SECTR_AudioEnvironment components, AudioEnvironmentTriggers can be overlapped and nested.

2.9 SECTR_AudioEnvironmentZone Class Reference

Activates a SECTR_AudioAmbience whenever a player enters an AudioReverbZone.

Inheritance diagram for SECTR_AudioEnvironmentZone:



Public Attributes

- SECTR_AudioAmbience Ambience = new SECTR_AudioAmbience()

  The configuraiton of the ambient audio in this Reverb Zone.

Properties

- bool Active `[get]`

  Returns true if this AudioEnvironment has put its Ambience on the stack.

Detailed Description

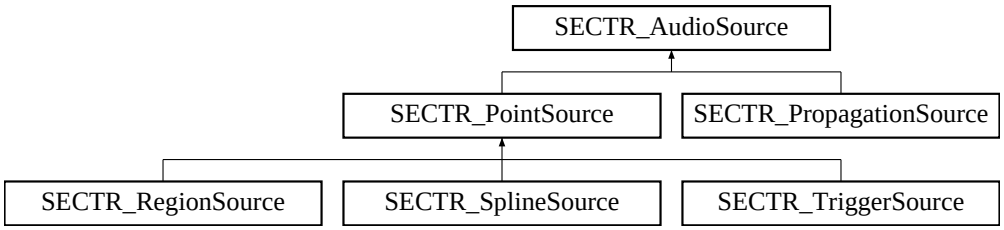
Activates a SECTR_AudioAmbience whenever a player enters an AudioReverbZone.

Audio Reverb can be an important part of creating a believeable Audio Environment. This component makes that easy, by ensuring that the specified Audio Environment is always active whenever the Reverb is audible. Because AudioReverbZone's are always spherical, the distance check is very inexpensive. As with all SECTR_AudioEnvironment components, AudioEnvironmentZones can be overlapped and nested.

2.10 SECTR_AudioSource Class Reference

An abstract base class for all components in SECTR Audio can be placed within the scene. AudioSource also provides a common interface to the user, and basic functions like play, stop, etc.

Inheritance diagram for SECTR_AudioSource:



Public Member Functions

- abstract void Play ()

  Make some noise! Plays the Cue.
- abstract void Stop (bool stopImmediately)

  Stops the Source from playing.
  Parameters

stopImmediately	*When true, overrides any fade out time set in the Cue.*

Public Attributes

- SECTR_AudioCue Cue = null

  The Cue to play from this source.
- bool Loop = true

  If the Cue should be forced to loop when playing.
- bool PlayOnStart = true

  Should the Cue auto-play when created.

Properties

- abstract bool IsPlaying `[get]`

  Returns true if the NoiseMaker is currently playing a sound.

Detailed Description

An abstract base class for all components in SECTR Audio can be placed within the scene. AudioSource also provides a common interface to the user, and basic functions like play, stop, etc.

It's important to note that AudioSource is not intendent to be the primary mechanism by which sounds are played, merely a convenient way to place sounds in the world, and in some cases trigger them from other built-in Unity features (like animation events). In generally, programmers wishing to play sounds based on game events, should so so by directly calling SECTR_AudioSystem.Play().

2.11 SECTR_AudioSystem Class Reference

The beating heart of SECTR_Audio, SECTR_AudioSystem provides all of the services necessary to play sounds and music, control the mix, etc.

Inherits MonoBehaviour.

Public Types

- enum OcclusionModes { OcclusionModes.Graph = 1 << 0, OcclusionModes.Raycast = 1 << 1, OcclusionModes.Distance = 1 << 2 }

  Flag set that determines which rules to use when computing audio Occlusion.

Static Public Member Functions

- static SECTR_AudioCueInstance Play (SECTR_AudioCue audioCue, Vector3 position, bool loop)

  Play an AudioCue at the specified position.
  Parameters

audioCue	*The SECTR_AudioCue to play.*
position	*The world space position at which to play the Cue.*
loop	*Forces this Cue to loop, even if it would not otherwise.*

  Returns

  A handle to the created instance.
- static SECTR_AudioCueInstance Play (SECTR_AudioCue audioCue, Transform parent, Vector3 localPosition, bool loop)

  Play an AudioCue at the specified position relative to the parent transform (if there is on).
  Parameters

audioCue	*The SECTR_AudioCue to play.*
parent	*An optional parent transform. If specified, position will be local to that transform.*
localPosition	*The world space position at which to play the Cue.*
loop	*Forces this Cue to loop, even if it would not otherwise.*

  Returns

  A handle to the created instance.
- static SECTR_AudioCueInstance Clone (SECTR_AudioCueInstance instance, Vector3 newPosition)

  Play an AudioCue at the specified position.
  Parameters

instance	*The SECTR_AudioCueInstance to duplicate.*
newPosition	*The world space position for the new instance.*

  Returns

  A handle to the created instance.
- static void PlayMusic (SECTR_AudioCue musicCue)

  Playes the specified Cue as music. Will soft-stop any currently playing music. Music should be 2D.

Parameters

musicCue	*The Cue to play.*

- static void StopMusic (bool stopImmediate)

  Stops the currently playing music.
  Parameters

stopImmediate	*If set to* `true` *stop immediate.*

- static void PushAmbience (SECTR_AudioAmbience ambience)

  Pushes the specified environment onto the stack of active envrionemnts. The item at the top of the stack will be audible.
  Parameters

ambience	*The ambience to add.*

- static void RemoveAmbience (SECTR_AudioAmbience ambience)

  Removes the specified environment from the stack of active envrionemnts. If it was at the top, the next highest will become active.
  Parameters

ambience	*The ambience to remove.*

- static void SetBusVolume (string busName, float volume)

  Sets user volume of the specified bus, if it exists. This is applied on top of whatever volume is set in the Bus resource.
  Parameters

busName	*The name of the bus to mute.*
volume	*The volume level.*

- static void SetBusVolume (SECTR_AudioBus bus, float volume)

  Sets user volume of the specified bus, if it exists. This is applied on top of whatever volume is set in the Bus resource.
  Parameters

bus	*The bus object to mute.*
volume	*The volume level.*

- static void MuteBus (string busName, bool mute)

  Sets the mute state of the specified bus, if it exists.
  Parameters

busName	*The name of the bus to mute.*
mute	*Mute on or off.*

- static void MuteBus (SECTR_AudioBus bus, bool mute)

  Sets the mute state of the specified bus.
  Parameters

bus	*The bus object to mute.*
mute	*Mute on or off.*

- static void PauseBus (string busName, bool paused)

  (Un)pauses the specified bus, if it exists.
  Parameters

busName	*The name of the bus to pause.*
paused	*Pause or unpause.*

- static void PauseBus (SECTR_AudioBus bus, bool paused)

  (Un)pauses the specified bus.
  Parameters

bus	*The bus object to pause.*
paused	*Pause or unpause.*

Public Attributes

- int MaxInstances = 128

  The maximum number of instances that can be active at once. Inaudible sounds do not count against this limit.
- int LowpassInstances = 32

  The number of instances to allocate with lowpass effects (for occlusion and the like).
- SECTR_AudioBus MasterBus = null

  The Bus at the top of the mixing heirarchy. Required to play sounds.
- SECTR_AudioAmbience DefaultAmbience = new SECTR_AudioAmbience()

  The baseline settings for any environmental audio. Will be audible when no other ambiences are active.
- float HDRBaseLoudness = 50

  Minimum Loudness for the HDR mixer. Current Loudness will never drop below this.
- float HDRWindowSize = 50

  The maximum difference between the loudest sound and the softest sound before sounds are simply culled out.
- float HDRDecay = 1

  Speed at which HDR window decays after a loud sound is played.
- bool BlendNearbySounds = true

  Should sounds close to the listener be blended into 2D (to avoid harsh stereo switching).
- Vector2 NearBlendRange = new Vector2(0.25f, 0.75f)

  Objects close to the listener will be blended into 2D, as a kind of fake HRTF. This determines the start and end of that blend.
- OcclusionModes OcclusionFlags = 0

  Determines what kind of logic to use for computing sound occlusion.
- float OcclusionDistance = 100f

  The distance beyond which sounds will be considered occluded, if Distance occlusion is enabled.
- LayerMask RaycastLayers = Physics.DefaultRaycastLayers

  The layers to test against when raycasting for occlusion.
- float OcclusionVolume = 0.5f

  The amount by which to decrease the volume of occluded sounds.
- float OcclusionCutoff = 2200

  The frequency cutoff of the lowpass filter for occluded sounds.
- float OcclusionResonanceQ = 1

  The resonance Q of the lowpass filter for occluded sounds.
- Vector2 RetestInterval = new Vector2(0.5f, 1f)

  The amount of time between tests to see if looping sounds should start or stop running.
- float CullingBuffer = 10f

  The amount of buffer to give before culling distant sounds.
- bool ShowAudioHUD = false

  Enable or disable of the in-game audio HUD.
- bool Debugging = false

  In the editor only, puts the listener at the AudioSystem, not at the Scene Camera.

Properties

- static bool Initialized `[get]`

  Returns true if there is an active AudioSystem in the scene.
- static SECTR_Member Member `[get]`

  Quick accessor for the Member of the Audio System.
- static SECTR_AudioSystem System `[get]`

  Quick accessor for the active AudioSystem.
- static Transform Listener `[get]`

  Accessor for the Listener, which has different behavior in game and in the editor.

Detailed Description

The beating heart of SECTR_Audio, SECTR_AudioSystem provides all of the services necessary to play sounds and music, control the mix, etc.

The most fundamental service AudioSystem provides is the ability to play sounds. Under the hood, the AudioSystem uses standard Unity AudioSources to play sounds, but layers on a number of signficant optimizations including object pooling, pre-culling of one shots, virtual instances of distant looping objects, and more. In aggregate, they provide a feature rich, but very high performance, solution for playing audio in Unity.

AudioSystem also manages the bus hierarchy, which can be used by designers to mix the game, and by programmers to dynamically modify volumes in response to user input or in-game events. Each AudioSystem instance must have a SECTR_AudioBus assigned to its MasterBus attribute, but it does not need to be the same Bus resource for every scene in the game. If desired, a game may have different bus hierarchies for different parts of the game.

Another useful service provided by the AudioSystem is the management and playback of SECTR_AudioEnvironments. AudioEnvironments are a powerful tool for establishing the basic sonic character of a part of the game world. While the AudioSystem only allows one AudioEnvironment to be active at a time, they are stored in a stack (where the topmost element is the highest priority cue. This interface allows clients (usually trigger volume type objects) to overlap and even be nested within one another, allowing sound designers to create rich, layered sonic spaces.

Lastly, the AudioSystem provides a simple interface for playing Music. Music in this case is simply a looping, 2D cue, but the system will ensure that there is only one "music" cue every playing at once (aside from cross fades between Cues). This simple concept maps well to the music implementations of most games, especially when combined with the playback options in SECTR_AudioCue. Future versions of SECTR_Audio may further extend the feature set of music playback.

Member Enumeration Documentation

enum SECTR_AudioSystem.OcclusionModes

Flag set that determines which rules to use when computing audio Occlusion.

Enumerator

Graph Uses the Sector/Portal graph to compute occlusion. Sound is occluded if it passes through a Closed Portal.

Raycast Uses the raycasts to compute occlusion. Sound is occluded if it passes through a collider.

Distance Uses the distance to compute occlusion. Sound is occluded if it is more than a certain distance from the listener.

2.12 SECTR_CharacterAudio Class Reference

Plays audio based on character events.

Inherits MonoBehaviour.

Public Attributes

- SurfaceSound DefaultSounds = new SurfaceSound()

  Default sounds to play if there is no material specific sound.
- List< SurfaceSound > SurfaceSounds = new List<SurfaceSound>()

  List of surface specific sounds.

Detailed Description

Plays audio based on character events.

2.13 SECTR_CharacterMotor Class Reference

C# adaptation of the Unity sample CharacterMotor, with custom tweaks and extensions.

Inherits MonoBehaviour.

Public Attributes

- CharacterMotorMovement movement = new CharacterMotorMovement()

  Basic movement properties.
- CharacterMotorJumping jumping = new CharacterMotorJumping()

  Jump specific movement properties.
- CharacterMotorMovingPlatform movingPlatform = new CharacterMotorMovingPlatform()

  Platform specific movment properties.

Detailed Description

C# adaptation of the Unity sample CharacterMotor, with custom tweaks and extensions.

2.14 SECTR_Chunk Class Reference

Chunk is the loadable/streamable version of a SECTR_Sector. The Chunk manages loading and unloading that data, usually at the request of a Loader component.

Inherits MonoBehaviour.

Public Member Functions

- void AddReference ()

  Add a reference to this Chunk. If this is the first reference, the data associated with the SectorChunk will be loaded. If you call AddReference, make sure to eventually call RemoveReference.
- void RemoveReference ()

  Add a reference to this Chunk. If this is the first reference, the data associated with the SectorChunk will be loaded.
- bool IsLoaded ()

  Determines whether the Chunk data is currently loaded.

  Returns

  True if this instance is loaded; otherwise false.
- bool IsUnloaded ()

  Determines whether the Chunk data is currently unloaded.

  Returns

  True if this instance is unloaded; otherwise false.
- float LoadProgress ()

  Returns the progress of the load, perhaps for use in an in-game display.

  Returns

  The progress as a float between 0 and 1.

Public Attributes

- string ScenePath

  The path of the scene to load.
- string NodeName

  The unique name of the root object in the exported Sector.
- bool ExportForReuse = false

  Exports the Chunk in a way that allows it to be shared by multiple Sectors, but may take more CPU to load.
- Mesh ProxyMesh

  A mesh to display when this Chunk is unloaded. Will be hidden when loaded.
- Material[] ProxyMaterials

  The per-submesh materials for the proxy.

Properties

- SECTR_Sector Sector `[get]`

  Returns the Sector associated with this Chunk.

Events

- LoadCallback Changed

  Event handler for load/unload callbacks.

Detailed Description

Chunk is the loadable/streamable version of a SECTR_Sector. The Chunk manages loading and unloading that data, usually at the request of a Loader component.

Chunk stores the data needed to load (and unload) a Sector that has been exported into a separate scene file. Loading will happen asynchronously if the user has a Pro license, synchronously otherwise.

Chunk uses a reference counted loading scheme, so multiple clients may safely request loading the same Chunk, provided that they equally match their Load requests with their Unload requests. Data for the Sector will be loaded when the reference count goes up from 0, and unloaded when it returns to 0.

2.15 SECTR_ChunkRef Class Reference

Allows for faster finding of loaded chunks.

Inherits MonoBehaviour.

Detailed Description

Allows for faster finding of loaded chunks.

2.16 SECTR_ComputeRMS Class Reference

Internal class to compute per-second RMS values of sounds and store them in HDR keys.

Inherits MonoBehaviour.

Properties

- float Progress [get]

  Returns the progress of the current bake, from 0 to 1.

Detailed Description

Internal class to compute per-second RMS values of sounds and store them in HDR keys.

2.17 SECTR_Culler Class Reference

Vestigial component from older version of SECTR. Left intact only for backwards compatability.

Inherits MonoBehaviour.

Public Attributes

- bool CullEachChild = false

  Overrides the culling information on Member.

Detailed Description

Vestigial component from older version of SECTR. Left intact only for backwards compatability.

2.18 SECTR_CullingCamera Class Reference

CullingCamera is the workhorse of SECTR Vis, culling objects by propagating Camera data down through the Sector/Portal graph and into individual SECTR_Culler objects.

Inherits MonoBehaviour.

Public Member Functions

- void ResetStats ()

  Resets all stats. Useful for demos.

Public Attributes

- bool SimpleCulling = false

  Forces culling into a mode designed for 2D and iso games where the camera is always outside the scene.
- float GizmoDistance = 10f

  Distance to draw clipped frustums.
- Material GizmoMaterial = null

  Material to use to render the debug frustum mesh.
- bool CullInEditor = false

  Makes the Editor camera display the Game view's culling while playing in editor.
- bool CullShadows = true

  Set to false to disable shadow culling post pass.
- Camera cullingProxy = null

Use another camera for culling properties.

- int NumWorkerThreads = 0

  Number of worker threads for culling. Do not set this too high or you may see hitching.

Properties

- static List< SECTR_CullingCamera > All `[get]`

  Returns a list of all enabled CullingCameras.
- Camera CullingCamera `[set]`

  Accessor that allows setting an override/proxy camera that will be used for culling instead of the camera on this object.
- int RenderersCulled `[get]`

  Return the number of renderers culled last frame.
- int LightsCulled `[get]`

  Return the number of lights culled last frame.
- int TerrainsCulled `[get]`

  Return the number of lights culled last frame.

Detailed Description

CullingCamera is the workhorse of SECTR Vis, culling objects by propagating Camera data down through the Sector/Portal graph and into individual SECTR_Culler objects.

Culling in SECTR is a fairly straightforward process. Each CullingCamera is expected to have a sibling Unity Camera. This allows PreCull() to be called, which is where our Camera does its work. CullingCamera cleans up after itself in PostRender, which allows multiple SECTR Cameras to be active in a single scene at once (if so desired).

Culling starts with the Sector(s) that contain the current Camera. From there, the CullingCamera walks the Sector graph. At each Portal, the Camera tests to see if its view frustum intersects the Portal's geometry. If it does, the frustum is clipped down by the Portal geometry, and the traversal continues to the next Sector (in a depth-first manner). Eventually, the frustum is winowed down to the point where no additional Portals are visible and the traversal completes.

SECTR Vis also allows the use of culling via instances of SECTR_Occluder. As the CullingCamera walks the Sector/Portal graph, it accumulates any Occluders that are present in that Sector. All future objects are then tested against the accumulated Occluders.

Lastly, shadow casting lights are accumulated during the traversal. Because of the complexities of shadow casting lights effectively extending the bounds of shadow casting meshes into Sectors that they would not otherwise occupy, the CullingCamera accumulates shadow casting point lights during the main traversal and then performs a post-pass for on any relevant meshes to ensure shadows are never prematurely culled.

2.19 SECTR_DemoUI Class Reference

A simple harness for demo messages and input handling.

Inherits MonoBehaviour.

Inherited by SECTR_AudioDemoUI, SECTR_CompleteDemoUI, SECTR_StreamDemoUI, and SECTR_VisDemoUI.

Public Attributes

- Texture2D Watermark

  Texture to display as a watermark.

- SECTR_GhostController PipController

  Link to a controllable ghost/spectator camera.
- string DemoMessage

  Message to display at start of demo.
- bool CaptureMode = false

  Disables HUD for video captures.

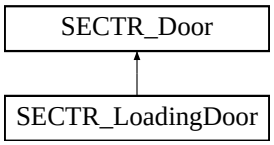
Detailed Description

A simple harness for demo messages and input handling.

2.20 SECTR_Door Class Reference

Implements a basic door component that is Portal aware. Also, provides an interface that more complex doors can implement.

Inheritance diagram for SECTR_Door:



Public Member Functions

- void OpenDoor ()

  Opens the door. Exposed for use by other script classes.
- void CloseDoor ()

  Closes the door. Exposed for use by other script classes.

Public Attributes

- SECTR_Portal Portal = null

  The portal this door affects (if any).
- string ControlParam = "Open"

  The name of the control param in the door.
- string CanOpenParam = "CanOpen"

  The name of the control param that indicates if we are allowed to open.
- string OpenState = "Base Layer.Open"

  The full name (layer and state) of the Open state in the Animation Controller.
- string ClosedState = "Base Layer.Closed"

  The full name (layer and state) of the Closed state in the Animation Controller.
- string OpeningState = "Base Layer.Opening"

  The full name (layer and state) of the Opening state in the Animation Controller.
- string ClosingState = "Base Layer.Closing"

  The full name (layer and state) of the Closing state in the Animation Controller.
- string WaitingState = "Base Layer.Waiting"

  The full name (layer and state) of the Wating state in the Animation Controller.

Detailed Description

Implements a basic door component that is Portal aware. Also, provides an interface that more complex doors can implement.

This door contains two base states (Open and Closed) and two transitional states (Opening and Closing). The animations for Open and Closed should be Looping animations, with one-shot animations for the transitions.

Door supports an optional reference to a SECTR_Portal. If set, the Door will manage the Closed flag of the Portal, which other systems will find useful.

2.21 SECTR_DoorAudio Class Reference

Extends the basic SECTR_Door with sounds that play on state transitions.

Inherits MonoBehaviour.

Public Attributes

- SECTR_AudioCue OpenLoopCue = null

  Sound to play while door is in Open state.
- SECTR_AudioCue ClosedLoopCue = null

  Sound to play while door is in Closed state.
- SECTR_AudioCue OpeningCue = null

  Sound to play when door starts to open.
- SECTR_AudioCue ClosingCue = null

  Sound to play while door starts to close.
- SECTR_AudioCue WaitingCue = null

  Sound to play while waiting for the door to start opening.

Detailed Description

Extends the basic SECTR_Door with sounds that play on state transitions.

There are four Cue's in this component, one for each state that the door can be in. Like the animations for the door, the open and closed Cues will be played looping, while the opening and closed cues are assumed to be one-shots.

2.22 SECTR_FogDisable Class Reference

Disables fog in the associated camera (for use in the PiP camera).

Inherits MonoBehaviour.

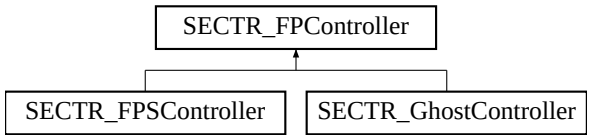
Detailed Description

Disables fog in the associated camera (for use in the PiP camera).

2.23 SECTR_FPController Class Reference

Simple abstract base class for first person style controllers.

Inheritance diagram for SECTR_FPController:



Public Attributes

- bool LockCursor = true

  Whether to lock the cursor when this camera is active.
- Vector2 Sensitivity = new Vector2(2f, 2f)

  Scalar for mouse sensitivity.
- Vector2 Smoothing = new Vector2(3f, 3f)

  Scalar for mouse smoothing.
- float TouchScreenLookScale = 1f

  Adjusts the size of the virtual joystick.

Detailed Description

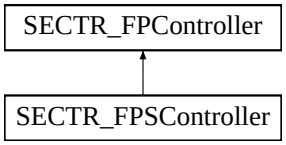
Simple abstract base class for first person style controllers.

This base class provides common services for FP style controllers, like translating both touch and mouse based inputs into camera rotation.

2.24 SECTR_FPSController Class Reference

A simple FPS style character controller.

Inheritance diagram for SECTR_FPSController:



Public Attributes

- bool LockCursor = true

  Whether to lock the cursor when this camera is active.
- Vector2 Sensitivity = new Vector2(2f, 2f)

  Scalar for mouse sensitivity.
- Vector2 Smoothing = new Vector2(3f, 3f)

  Scalar for mouse smoothing.
- float TouchScreenLookScale = 1f

  Adjusts the size of the virtual joystick.

Detailed Description

A simple FPS style character controller.

Extends the FP Controller to translate input input into movement that a Character Motor can understand.

2.25 SECTR_Geometry Class Reference

A library of useful geometric functions.

Static Public Member Functions

- static Bounds ComputeBounds (Light light)

  Computes the bounds of the input Light. Area and Directional lights are treated as points as they have no good representation in SECTR.
  Parameters

light	*The light whose bounds need computing.*

  Returns

  The world space Bounds of the light.
- static Bounds ComputeBounds (Terrain terrain)

  Computes the bounds of the input Terrain.
  Parameters

terrain	*The terrain whose bounds need computing.*

  Returns

  The world space Bounds of the terrain.
- static bool FrustumIntersectsBounds (Bounds bounds, List< Plane > frustum, int inMask, out int outMask)

  Determines if an AABB intersects a frustum.
  Parameters

bounds	*The AABB to check for inclusion.*
frustum	*An array of planes that define the frustum.*
inMask	*A bitmask of which planes to test for intersection, as computed by a parent AABB.*
outMask	*The bitmask of planes that intersect this AABB.*

  Returns

  Returns true if it is fully or partially contained, false otherwise.
- static bool FrustumContainsBounds (Bounds bounds, List< Plane > frustum)

  Determines if an AABB is fulling contained within a frustum.
  Parameters

bounds	*The AABB to check for inclusion.*
frustum	*An array of planes that define the frustum.*

  Returns

  Returns true if it is fully contained, false otherwise.
- static bool BoundsContainsBounds (Bounds container, Bounds contained)

  Tests to see if one AABB fully contains another AABB.
  Parameters

container	*The AABB that does the containing.*
contained	*The AABB to test for containment.*

  Returns

  Returns true if the AABB is fully contained.
- static bool BoundsIntersectsSphere (Bounds bounds, Vector3 sphereCenter, float sphereRadius)

  Tests to see if one an AABB and a Sphere intersect.
  Parameters

bounds	*The AABB for the test.*
sphereCenter	*The center of the sphere.*
sphereCenter	*The center of the sphere.*
sphereRadius	*The radius of the sphere.*

Returns

Returns true if the AABB and Sphere intersect.

- static Bounds ProjectBounds (Bounds bounds, Vector3 projection)

  Extrudes an AABB along a ray.
  Parameters

bounds	*The original AABB.*
projection	*The direction and distance by which to project the bounds.*

  Returns

  The extruded AABB.

- static bool IsPointInFrontOfPlane (Vector3 position, Vector3 center, Vector3 normal)

  Determines if a point is in front of or behind a plane.
  Parameters

position	*The position of the point.*
center	*A point on the plane.*
normal	*The normal of the plane.*

- static bool IsPolygonConvex (Vector3[] verts)

  Determines if is polygon convex. Verts must be sorted in CW or CCW order.
  Parameters

verts	*The sorted array of verts in the polygon.*

  Returns

  True if the polygon is convex. False otherwise.

- static int CompareVectorsCW (Vector3 a, Vector3 b, Vector3 centroid, Vector3 normal)

  Determines the relative order of two points on a plane.
  Parameters

a	*The first position to compare.*
b	*The second position to compare.*
centroid	*The centroid of the reference polygon.*
normal	*The normal about which to measure the "rotation".*

  Returns

  1 if they are CW, -1 for CCW, and 0 if they are identical.

Detailed Description

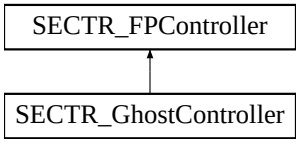
A library of useful geometric functions.

SECTR is an inherently geometric library, and this class is a common repository for useful geometric methods shared by several classes in the library.

2.26 SECTR_GhostController Class Reference

Implements a standard spectator/fly camera.

Inheritance diagram for SECTR_GhostController:



Public Attributes

- float FlySpeed = 0.5f

  The speed at which to fly through the world.
- float AccelerationRatio = 1f

  The translation acceleration amount applied by keyboard input.
- float SlowDownRatio = 0.5F

  The amount by which holding down Ctrl slows you down.
- bool LockCursor = true

  Whether to lock the cursor when this camera is active.
- Vector2 Sensitivity = new Vector2(2f, 2f)

  Scalar for mouse sensitivity.
- Vector2 Smoothing = new Vector2(3f, 3f)

  Scalar for mouse smoothing.
- float TouchScreenLookScale = 1f

  Adjusts the size of the virtual joystick.

Detailed Description

Implements a standard spectator/fly camera.

Simple class adds movement to the FP Controller base. Useful for debug cameras and the like.

2.27 SECTR_Graph Class Reference

A set of static utility functions used to traverse the Sector/Portal graph.

Classes

- class Node

  Represents a Node in the Sector/Portal graph. Contains useful data for implementing traversals.

Static Public Member Functions

- static void DepthWalk (ref List< Node > nodes, SECTR_Sector root, SECTR_Portal.PortalFlags stopFlags, int maxDepth)

  Generates a List of nodes that is a depth-first traversal of walk of sector graph from the specified root.

  Parameters

nodes	*List into which walk results will be written.*
root	*The Sector at which to start the traversal.*
stopFlags	*Flag set to test at each SECTR_Portal. A failed test will stop the traversal.*
maxDepth	*The depth into the graph at which to end the traversal. -1 means no limit.*

Returns

A List of Nodes in depth-first traveral order.

- static void BreadthWalk (ref List< Node > nodes, SECTR_Sector root, SECTR_Portal.PortalFlags stopFlags, int maxDepth)

  Generates a List of nodes that is a braedth-first traversal of walk of sector graph from the specified root.
  Parameters

nodes	*List into which walk results will be written.*
root	*The Sector at which to start the traversal.*
stopFlags	*Flag set to test at each SECTR_Portal. A failed test will stop the traversal.*
maxDepth	*The depth into the graph at which to end the traversal. -1 means no limit.*

  Returns

  A List of Nodes in breadth-first traveral order.

- static void FindShortestPath (ref List< Node > path, Vector3 start, Vector3 goal, SECTR_Portal.PortalFlags stopFlags)

  Finds the shortest path through the portal graph between two points. The start and end points must currently be within Sector in the graph.
  Parameters

path	*List into which search results will be written.*
start	*The world space position at which to start the search.*
goal	*The world space goal of the search.*
stopFlags	*Flag set to test at each SECTR_Portal. A failed test will stop the traversal.*

  Returns

  A list of nodes from the Start to the Goal. Empty if there is no path.

- static string GetGraphAsDot (string graphName)

  Gets the graph as dot formatted string (for visualization in GraphViz and the like).
  Parameters

graphName	*The name to embed in the graph file.*

  Returns

  The graph as dot formatted string.

Detailed Description

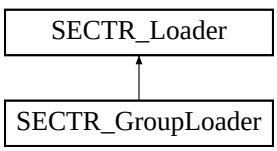
A set of static utility functions used to traverse the Sector/Portal graph.

The set of Sectors and Portals can be thought of as a graph, where the Sectors are the nodes and the Portals are the edges. SectorGraph implements useful funtions for traversing or otherwise searching the graph.

2.28 SECTR_GroupLoader Class Reference

Allows users to group a set of Sectors, loading and unloading them as if they were a single Sector.

Inheritance diagram for SECTR_GroupLoader:



Public Attributes

- List< SECTR_Sector > Sectors = new List<SECTR_Sector>()

The Sectors to load and unload together.

Properties

- override bool Loaded `[get]`

  Returns true if all Sectors in the group are loaded.

Detailed Description

Allows users to group a set of Sectors, loading and unloading them as if they were a single Sector.

There are occasions where a section of the scene needs to be split into multiple Sectors (perhaps for occlusion culling or game logic) but they need be loaded as if they were part of a single Sectors. Group Loader takes care of this, by automatically incrementing and decrementing reference counts whenever one of the Sectors in the list is loaded or unloaded.

2.29 SECTR_Hibernator Class Reference

Automatically enables and disables components on itself when the SECTR_Sector it's part of are (un)loaded.

Inherits MonoBehaviour.

Public Member Functions

- delegate void HibernateCallback ()

  Delegate delcaration for anyone who wants to be notified on hibernation related events.

Public Attributes

- bool HibernateChildren = true

  Hibernate components on children as well as ones on this game object.
- bool HibernateBehaviors = true

  Disable Behavior components during hibernation.
- bool HibernateColliders = true

  Disable Collder components during hibernation.
- bool HibernateRigidBodies = true

  Disable RigidBody components during hibernation.
- bool HibernateRenderers = true

  Hide Render components during hibernation.

Events

- HibernateCallback Awoke

  Event handler for when we go from hiberanted to awake.
- HibernateCallback Hibernated

  Event handler for when we go from awake to hibernate.
- HibernateCallback HibernateUpdate

  Event handler for updates during hibernation. Use judiciously.

Detailed Description

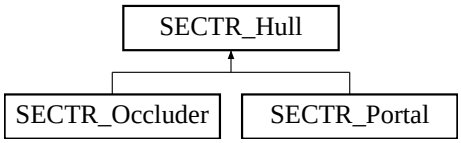
Automatically enables and disables components on itself when the SECTR_Sector it's part of are (un)loaded.

It's often useful to have global objects that are always instantiated, even when their part of the scene is not loaded. However, because their area is not loaded, these objets may not want to update until their areas are active again. SECTR_Hibernator takes care of this behavior automatically, taking care of physics and behaviors, while providing optional Events to notify anyone who might be interested in things that happen while hibernated.

2.30 SECTR_Hull Class Reference

Abstract base class that implements planar, convex hulls, for use in SECTR_Portal, SECTR_Occluder and other client classes.

Inheritance diagram for SECTR_Hull:



Public Member Functions

- bool IsPointInHull (Vector3 p, float distanceTolerance)

  Determines whether the given point is inside the extents of the hull. Distance tolerance will reject points more than that distance from the plane.

  Parameters

p	*The point to test.*
distanceTolerance	*The maximum distance to be considered "in the hull".*

Public Attributes

- Mesh HullMesh = null

  Convex, planar mesh that defines the portal shape.

Properties

- Vector3[] VertsCW `[get]`

  Returns the verts in clockwise order.
- Vector3 Normal `[get]`

  Returns the world space normal of the Hull.
- Vector3 ReverseNormal `[get]`

  Returns the world space, backwards facing normal of the hull.
- Vector3 Center `[get]`

  Returns the world space centroid of the Hull.
- Plane HullPlane `[get]`

  Returns the world space plane of this hull.
- Plane ReverseHullPlane `[get]`

  Returns the world space plane of this hull, but with the normal flipped.

Detailed Description

Abstract base class that implements planar, convex hulls, for use in SECTR_Portal, SECTR_Occluder and other client classes.

Planar, convex hulls are a common pattern within the framework. They provide a reasonable balance between CPU cost and versatility. In order to allow geometry to be created within Unity or using external modelling programs, Hulls are based on standard Unity Mesh resources, and are lazily converted into a simpler, loop representation at runtime.

2.31 SECTR_ImpactAudio Class Reference

Playes a SECTR_AudioCue when a physics impact is detected.

Inherits MonoBehaviour.

Public Attributes

- ImpactSound DefaultSound = null

  Default sound to play on impact.
- List< ImpactSound > SurfaceImpacts = new List<ImpactSound>()

  Surface specific impact sounds.
- float MinImpactSpeed = .01f

  The minimum relative speed at the time of impact required to trigger this cue.
- float MinImpactInterval = 0.5f

  The minimum amount of time between playback of this sound.

Detailed Description

Playes a SECTR_AudioCue when a physics impact is detected.

ImpactSource supports any collider that Unity allows, provided it's setup to create and recieve collision.

2.32 SECTR_LightmapRef Class Reference

Stores the references to lightmap textures in an exported Chunk.

Inherits MonoBehaviour.

Static Public Member Functions

- static void InitRefCounts ()

  Initializes the global/static lightmap ref count array. Can be called multiple times, but should only be called at the start of the level and only by SECTR_Chunk.

Properties

- List< RefData > LightmapRefs `[get]`

  Read-only accessor for the LightmapRefs. Intended primarily for debugging, and fixup during imports.

Detailed Description

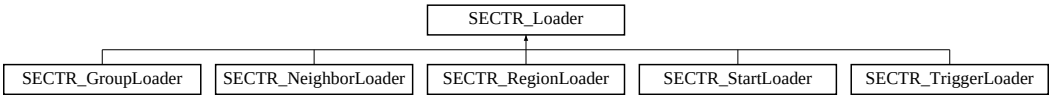
Stores the references to lightmap textures in an exported Chunk.

This class is meant for internal use only and should not be added by users, though every attempt has been made to ensure nothing bad happens in the case that one is added accidentally.

2.33 SECTR_Loader Class Reference

Provides an abstract base class for classes that load data from SECTR_Chunk components.

Inheritance diagram for SECTR_Loader:



Properties

- abstract bool Loaded `[get]`

  Returns true if all referenced Chunks are loaded. False, otherwise.

Detailed Description

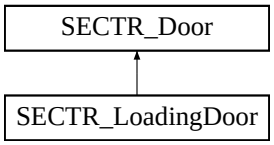
Provides an abstract base class for classes that load data from SECTR_Chunk components.

Classes are not required to derive from SECTR_Loader in order to (un)load chunk data. This class merely provides common functionality useful in many of the built in Loaders.

2.34 SECTR_LoadingDoor Class Reference

Extends the basic SECTR_Door with awareness of streaming SECTR_Chunks. This door won't open unless the SECTR_Chunks on both sides of the door's SECTR_Portal are loaded.

Inheritance diagram for SECTR_LoadingDoor:



Public Member Functions

- void OpenDoor ()

  Opens the door. Exposed for use by other script classes.
- void CloseDoor ()

  Closes the door. Exposed for use by other script classes.

Public Attributes

- LayerMask LoadLayers = (int)0xffffff

  Specifies which layers are allow to cause loads (vs simply opening the door).
- bool FadeBeforeLoad = false

  Should screen fade to black before loading.
- float FadeTime = 1f

  How long to fade out before loading. Also, how long to fade back in.
- float HoldTime = 0.1f

  How long to stay faded out. Helps cover pops right at the moment of loading.
- Color FadeColor = Color.black

  The color to fade the screen to on load.
- SECTR_Portal Portal = null

  The portal this door affects (if any).
- string ControlParam = "Open"

  The name of the control param in the door.
- string CanOpenParam = "CanOpen"

  The name of the control param that indicates if we are allowed to open.
- string OpenState = "Base Layer.Open"

  The full name (layer and state) of the Open state in the Animation Controller.
- string ClosedState = "Base Layer.Closed"

  The full name (layer and state) of the Closed state in the Animation Controller.
- string OpeningState = "Base Layer.Opening"

  The full name (layer and state) of the Opening state in the Animation Controller.
- string ClosingState = "Base Layer.Closing"

  The full name (layer and state) of the Closing state in the Animation Controller.
- string WaitingState = "Base Layer.Waiting"

  The full name (layer and state) of the Wating state in the Animation Controller.

Detailed Description

Extends the basic SECTR_Door with awareness of streaming SECTR_Chunks. This door won't open unless the SECTR_Chunks on both sides of the door's SECTR_Portal are loaded.

Unity restricts their async APIs to Pro owners, which means that when a Chunk is loaded, it may cause a noticeable hitch for non-Pro users. This component is an example of how to hide that hitch.

2.35 SECTR_LOD Class Reference

Implements a simple Level of Detail (LOD) system for SECTR objects.

Inherits MonoBehaviour.

Public Types

- enum SiblingIFlags { SiblingIFlags.Behaviors = 1 << 0, SiblingIFlags.Renderers = 1 << 1, SiblingIFlags.-Lights = 1 << 2 }

  The set of bitflags that determine which siblings, if any are disabled when the LOD is culled.

Public Member Functions

- void SelectLOD (Camera renderCamera)

  Picks the correct LOD based on the specified camera.

Parameters

renderCamera	*The camera for which to select the LOD.*

Public Attributes

- List< LODSet > LODs = new List<LODSet>()

  This list of LOD sets for this object.
- SiblingIFlags CullSiblings = 0

  When set to true disables sibling mono behaviors, renderers and lights when the system is culled. Determines which sibling components are disabled when the LOD is culled.

Properties

- static List< SECTR_LOD > All `[get]`

  Accessor for global list of active SECTR_LOD components.

Detailed Description

Implements a simple Level of Detail (LOD) system for SECTR objects.

LOD in SECTR is based on the size of the object bounds in screen space, which is the same metric as the LOD system in Unity Pro. SECTR LODs may have as many LOD objects as desired, and can affect any game object, not just renderers. LOD requires a SECTR_CullingCamera in the scene for the LODs to be updated.

Member Enumeration Documentation

enum SECTR_LOD.SiblingIFlags

The set of bitflags that determine which siblings, if any are disabled when the LOD is culled.

Enumerator

Behaviors Disable sibling behaviors that are not this component or the member.

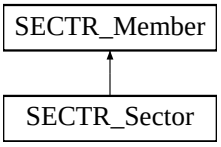
Renderers Disable sibling members.

Lights Disable sibling lights.

2.36 SECTR_Member Class Reference

Member represents anything that can be part of a SECTR_Sector, including Sectors themselves.

Inheritance diagram for SECTR_Member:



Classes

- class Child

  Simple data structure to represent the important information about one of the children of a SECTR_Member.

Public Types

- enum BoundsUpdateModes { BoundsUpdateModes.Start, BoundsUpdateModes.Movement, BoundsUpdateModes.Always, BoundsUpdateModes.Static }

  Rules for how often to update the children and bounds.
- enum ChildCullModes { ChildCullModes.Default, ChildCullModes.Group, ChildCullModes.Individual }

  Modes for how to cull children (VIS only)

Public Member Functions

- void ForceUpdate (bool updateChildren)

  Forces an update. Used only in special cases.
- void SectorDisabled (SECTR_Sector sector)

  Called when a Sector is being destroyed.
- delegate void MembershipChanged (List< SECTR_Sector > left, List< SECTR_Sector > joined)

  Delegate delcaration for anyone who wants to be notified when membership changes.

Public Attributes

- bool PortalDetermined = false

  Set to true if Sector membership should only change when crossing a portal.
- SECTR_Sector ForceStartSector = null

  If set, forces the initial Sector to be the specified Sector.
- BoundsUpdateModes BoundsUpdateMode = BoundsUpdateModes.Always

  Determines how often the bounds are recomputed. More frequent updates requires more CPU.
- float ExtraBounds = SECTR_Geometry.kBOUNDS_CHEAT

  Adds a buffer on bounding box to compensate for minor imprecisions.
- bool OverrideBounds = false

  Override computed bounds with the user specified bounds. Advanced users only.
- Bounds BoundsOverride

  User specified override bounds. Auto-populated with the current bounds when override is inactive.
- Light DirShadowCaster

  Optional shadow casting directional light to use in membership calculations. Bounds will be extruded away from light, if set.
- float DirShadowDistance = 100

  Distance by which to extend the bounds away from the shadow casting light.
- ChildCullModes ChildCulling = ChildCullModes.Default

  Determines if this SectorCuller should cull individual children, or cull all children based on the aggregate bounds.

Properties

- static List< SECTR_Member > All `[get]`

  Returns a list of all enabled Members.
- bool CullEachChild `[get]`

  Returns true if each child should be culled individually.
- List< SECTR_Sector > Sectors `[get]`

  Returns all the Sectors that this object belongs to.
- List< Child > Children `[get]`

  Returns a flattened list of all relevant children.
- List< Child > Renderers `[get]`

Returns the subset of the Children list that contains Renderers.

- bool ShadowCaster [get]

  Returns true if any child renderer components casts shadows.
- List< Child > ShadowCasters [get]

  Returns the subset of the Children list that cast shadows.
- List< Child > Lights [get]

  Returns the subset of the Children list that contains Lights.
- bool ShadowLight [get]

  Returns true if any child light components create shadows.
- List< Child > ShadowLights [get]

  Returns the subset of the Children list that create shadows.
- List< Child > Terrains [get]

  Returns the subset of the Children list that contains Lights.
- Bounds TotalBounds [get]

  Returns the union of the RenderBounds and LightBounds.
- Bounds RenderBounds [get]

  Returns the union of the Bounds of all child Renderers.
- bool HasRenderBounds [get]

  Returns true if the RenderBounds contains valid data.
- Bounds LightBounds [get]

  Returns the union of the Bounds of all child Lights.
- bool HasLightBounds [get]

  Returns true if the LightBounds contains valid data.
- bool Frozen [get, set]

  (Un)Freezes (i.e. disables updates and preserves bounds).
- SECTR_Member ChildProxy [set]

  Setting the ChildProxy will force this Member to use the children of another node as if they were its own. Only to be used in very specific circumstances.
- bool NeverJoin [set]

  Setting to true will prevent this Member from joining any sectors. Only to be used in very specific circumstances.
- bool IsSector [get]

  Returs if this Member is really a Sector.

Events

- MembershipChanged Changed

  Event handler for membership changed callbacks.

Detailed Description

Member represents anything that can be part of a SECTR_Sector, including Sectors themselves.

Member's primary job is to figure out which Sectors a given GameObject belongs to. In order to accomplish this goal, it needs to compute a bounding box (actually several) and to periodically check to see which Sectors overlap that box. SectorMember also caches information that other clients are interested in, such as the aggregate render bounds of its children, or a list of child Light components.

Members may be dynamic or static, and will save some per-frame CPU work if marked as static. Care should be taken that static Members by accurately marked, as moving children of a static Member may produce unexpected results.

Note that Members may have children that are themselves Members. The code will automatically detect this case and not include the child Member's children in its list. This can be a very convienient way to control the granuality of scene partitioning, especially for culling and streaming.

Member Enumeration Documentation

enum SECTR_Member.BoundsUpdateModes

Rules for how often to update the children and bounds.

Enumerator

Start Compute children on start. Update bounds on movement.

Movement Compute children and bounds on movement.

Always Compute children and bounds on every update.

Static Compute children and bounds on only on start.

enum SECTR_Member.ChildCullModes

Modes for how to cull children (VIS only)

Enumerator

Default Cull Sector children individually, Member children as a group.

Group Cull children as a group.

Individual Cull each child individually.

2.37 SECTR_MusicTrigger Class Reference

Makes the specified music active when a trigger is entered.

Inherits MonoBehaviour.

Public Attributes

- SECTR_AudioCue Cue = null

  The Cue to play as music. If null, this trigger will stop the current music.
- bool Loop = true

  Should music be forced to loop when playing.
- bool StopOnExit = false

  Should the music stop when leaving the trigger.

Detailed Description

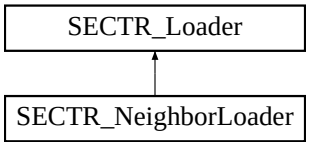
Makes the specified music active when a trigger is entered.

TriggerSource supports any collider that Unity allows, provided it's marked to be a trigger.

2.38 SECTR_NeighborLoader Class Reference

Loads SECTR_Chunk components that are in the current or adjacent SECTR_Sector.

Inheritance diagram for SECTR_NeighborLoader:



Public Attributes

- int MaxDepth = 1

  Determines how far out to load neighbor sectors from the current sector. Depth of 0 means only the current Sector.

Properties

- override bool Loaded `[get]`

  Returns true if all referenced Chunks are loaded. False, otherwise.

Detailed Description

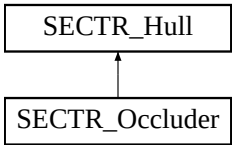
Loads SECTR_Chunk components that are in the current or adjacent SECTR_Sector.

Neighbor Loader determines which SECTR_Chunk objects to load/unload by performing a breadth-first walk of the Sector/Portal graph. The depth of this walk is limited by the MaxDepth property, where a value of zero means "only the current Sector", a depth of one means "current and each adjacent sector", etc. Many games will want to place a Neighbor Loader on the player to ensure that the Sectors they are in or near are always loaded.

2.39 SECTR_Occluder Class Reference

An Occluder represents a visual obstruction. It will hide any objects behind it (from the perspective of the current SECTR_Culler).

Inheritance diagram for SECTR_Occluder:



Public Types

- enum OrientationAxis {
  OrientationAxis.None, OrientationAxis.XYZ, OrientationAxis.XZ, OrientationAxis.XY,
  OrientationAxis.YZ }

  Possible axes for auto-orientation.

Public Member Functions

- Matrix4x4 GetCullingMatrix (Vector3 cameraPos)

  Returns the local to world matrix to be used to transform verts during culling.
- bool IsPointInHull (Vector3 p, float distanceTolerance)

  Determines whether the given point is inside the extents of the hull. Distance tolerance will reject points more than that distance from the plane.

Parameters

p	*The point to test.*
distanceTolerance	*The maximum distance to be considered "in the hull".*

Public Attributes

- OrientationAxis AutoOrient = OrientationAxis.None

  The axes that should orient towards the camera during culling (if any).
- Mesh HullMesh = null

  Convex, planar mesh that defines the portal shape.

Properties

- static List< SECTR_Occluder > All `[get]`

  Accessor for quickly retrieving all SectorOccluders.
- SECTR_Member Member `[get]`

  Fast access to the required SectorMember sibling.
- Vector3[] VertsCW `[get]`

  Returns the verts in clockwise order.
- Vector3 Normal `[get]`

  Returns the world space normal of the Hull.
- Vector3 ReverseNormal `[get]`

  Returns the world space, backwards facing normal of the hull.
- Vector3 Center `[get]`

  Returns the world space centroid of the Hull.
- Plane HullPlane `[get]`

  Returns the world space plane of this hull.
- Plane ReverseHullPlane `[get]`

  Returns the world space plane of this hull, but with the normal flipped.

Detailed Description

An Occluder represents a visual obstruction. It will hide any objects behind it (from the perspective of the current SECTR_Culler).

Occluders are a useful tool for optimizing culling, especially in outdoor scenes where Portals may be rare. Occluders are somewhat expensive, and should be used judiciously, ideally on very large objects which obstruct many objects behind them, not on many small objects.

Like Portals, Occluders are required to be planar, convex shapes. This constraint is satisfactory for many cases, however, it is often desirable that an occluder represent an object with a 3D volume, obstructing regardless of viewing angle. To efficiently allow this behavior, Occluders support an AutoOrient feature, that will automatically orient them towards the current SectorCuller during culling.

Member Enumeration Documentation

enum SECTR_Occluder.OrientationAxis

Possible axes for auto-orientation.

Enumerator

None No auto orientation.

XYZ Orient all axes.

XZ Orient on world space XZ axes.

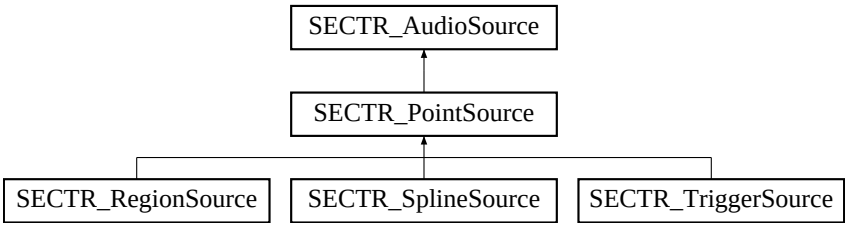
XY Orient on world space XY axes.

YZ Orient on world space YZ axes.

2.40 SECTR_PointSource Class Reference

Plays a SECTR_AudioCue at this point in the world.

Inheritance diagram for SECTR_PointSource:



Public Member Functions

- override void Play ()

  Make some noise! Plays the Cue.
- override void Stop (bool stopImmediately)

  Stops the Source from playing.
  Parameters

stopImmediately	*Overrides any fade-out specified in the Cue*

Public Attributes

- SECTR_AudioCue Cue = null

  The Cue to play from this source.
- bool Loop = true

  If the Cue should be forced to loop when playing.
- bool PlayOnStart = true

  Should the Cue auto-play when created.

Properties

- override bool IsPlaying `[get]`

  Returns true if the NoiseMaker is currently playing a sound.

Detailed Description

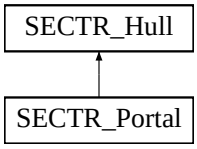
Plays a SECTR_AudioCue at this point in the world.

Point Source is the SECTR Audio equivalent of Unity's AudioSource component in that it simply plays a sound at a point in space. Point Source, however, benefits from the full set of creating, mixing, and other advanced features of SECTR Audio, but is only barely more expensive than a raw Unity AudioSource.

2.41 SECTR_Portal Class Reference

Portals define the logical and geometric connection between two SECTR_Sector objects.

Inheritance diagram for SECTR_Portal:



Public Types

- enum PortalFlags { PortalFlags.Closed = 1 << 0, PortalFlags.Locked = 1 << 1, PortalFlags.PassThrough = 1 << 2 }

  The set of bitflags that can be set on (and tested for) a given portal. Users are encouraged to add to this set, but after the Reserved range to avoid interference with future updates.

Public Member Functions

- IEnumerable< SECTR_Sector > GetSectors ()

  Creates an iterator that steps through all of the connected Sectors. Helpful way to make some iterating client code more generic.
- void SetFlag (PortalFlags flag, bool on)

  Sets a particular flag on this Portal to be true.
  Parameters

flag	*The flag to make true.*
on	*Sets the flag on or off.*

- bool IsPointInHull (Vector3 p, float distanceTolerance)

  Determines whether the given point is inside the extents of the hull. Distance tolerance will reject points more than that distance from the plane.
  Parameters

p	*The point to test.*
distanceTolerance	*The maximum distance to be considered "in the hull".*

Public Attributes

- PortalFlags Flags = 0

  Flags for this Portal. Used in graph traversals and the like.
- Mesh HullMesh = null

  Convex, planar mesh that defines the portal shape.

Properties

- static List< SECTR_Portal > All `[get]`

  Returns a list of all enabled portals.
- SECTR_Sector FrontSector `[get, set]`

  Accessor for the Sector link on the front side of the SectorPortal. When set, properly notifies the previous Sector of connection changes.
- SECTR_Sector BackSector `[get, set]`

Accessor for the Sector link on the back side of the SectorPortal. When set, properly notifies the previous Sector of connection changes.

- bool Visited `[get, set]`

  Utility property for tracking during graph walks.
- Vector3[] VertsCW `[get]`

  Returns the verts in clockwise order.
- Vector3 Normal `[get]`

  Returns the world space normal of the Hull.
- Vector3 ReverseNormal `[get]`

  Returns the world space, backwards facing normal of the hull.
- Vector3 Center `[get]`

  Returns the world space centroid of the Hull.
- Plane HullPlane `[get]`

  Returns the world space plane of this hull.
- Plane ReverseHullPlane `[get]`

  Returns the world space plane of this hull, but with the normal flipped.

Detailed Description

Portals define the logical and geometric connection between two SECTR_Sector objects.

If a Sector is a room then a Portal is like a window or doorway into or out of that room. Portals not only define which portals are connected to each other, but the shape and size of those connections. Portals can have as many vertices as necessary, but they must be convex and planar.

Like the rest of the system, Portals are completely dynamic, and can translate/rotate/scale at runtime without any performance penalty. They can also be turned on and off (i.e. when a door opens or closes) simply by setting their enabled flag on or off.

Member Enumeration Documentation

enum SECTR_Portal.PortalFlags

The set of bitflags that can be set on (and tested for) a given portal. Users are encouraged to add to this set, but after the Reserved range to avoid interference with future updates.

Enumerator

Closed Portal cannot be seen through.

Locked Portal is locked.

PassThrough Portal will be visible independent of geometry (but normal direction still matters).

2.42 SECTR_PriorityQueue< T > Class Template Reference

Implements a priority queue in terms of a binary heap.

Public Member Functions

- void Enqueue (T item)

  Enqueue the specified item.

Parameters

item	*The item to enqueue.*

- T Dequeue ()

  Dequeue the lowest priority item from the queue.
- T Peek ()

  Examine the lowest priority item but don't remove it.
- override string ToString ()

  Returns a nice string that represents the current state of the queue.
- bool IsConsistent ()

  Indicates if the queue is consistent/properly sorted.

Properties

- int Count `[get, set]`

  Returns the number of items in the queue.
- T this[int index] `[get, set]`

  Retrieves or modifies the item at the specified index.

Detailed Description

Implements a priority queue in terms of a binary heap.

Based on http://visualstudiomagazine.com/articles/2012/11/01/priority-queues-with-c.-aspx.

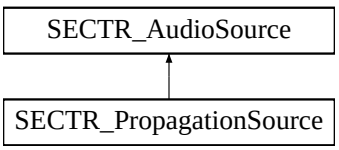
Type Constraints

T **:** ***IComparable*<*T*>**

2.43 SECTR_PropagationSource Class Reference

Propagation Source simulates the complex phenomena of audio reflections in a closed space.

Inheritance diagram for SECTR_PropagationSource:



Public Member Functions

- override void Play ()

  Make some noise! Plays the Cue.
- override void Stop (bool stopImmediately)

  Stops the Source from playing.

Parameters

stopImmediately	*Overrides any fade-out specified in the Cue*

Public Attributes

- float InterpDistance = 2f

  When the listener gets within this distance of a portal, the sound direction will start to blend towards the next portal or source position.
- SECTR_AudioCue Cue = null

  The Cue to play from this source.
- bool Loop = true

  If the Cue should be forced to loop when playing.
- bool PlayOnStart = true

  Should the Cue auto-play when created.

Properties

- override bool IsPlaying `[get]`

  Returns true if the Source is currently playing a sound.

Detailed Description

Propagation Source simulates the complex phenomena of audio reflections in a closed space.

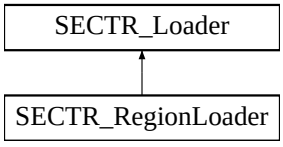
In the real world, the sounds we hear are very often reflections of the actual space. As such, the sound appears to be located not in a straight line to the source, but to be emanating from the nearest opening that leads to the source. The Sector/Portal graph provides the perfect context for efficiently but accurately determining how sounds move through an environment.

Propagation Sources works by attemtping to find the shortest path between itself and the active AudioListener. Because the Sector/Portal graph is fairly coarse, this path plan is relatively inexpensive, but it is not free. Because of this additional cost, Propagation Sources should be used sparingly, where they provide the most audio bang for your CPU buck.

2.44 SECTR_RegionLoader Class Reference

(Un)loads Chunks within a given volume. Can be set to optionally not touch Sectors that are not part of the terrain grid.

Inheritance diagram for SECTR_RegionLoader:



Public Attributes

- Vector3 LoadSize = new Vector3(20f, 10f, 20f)

  The dimensions of the volume in which terrain chunks should be loaded.
- float UnloadBuffer = 0.1f

The distance from the load size that you need to move for a Sector to unload (as a percentage).

- LayerMask LayersToLoad = -1

  If set, will only load Sectors in matching layers.

Properties

- override bool Loaded `[get]`

  Returns true if all referenced Chunks in region are loaded. False, otherwise.

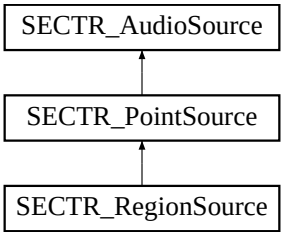
Detailed Description

(Un)loads Chunks within a given volume. Can be set to optionally not touch Sectors that are not part of the terrain grid.

2.45 SECTR_RegionSource Class Reference

Playes a SECTR_AudioCue within a 3D volume.

Inheritance diagram for SECTR_RegionSource:



Public Member Functions

- override void Play ()

  Make some noise! Plays the Cue.

- override void Stop (bool stopImmediately)

  Stops the Source from playing.
  Parameters

stopImmediately	*Overrides any fade-out specified in the Cue*

Public Attributes

- bool Raycast = false

  Determine the closest point by raycast instead of bounding box. More accurate but more expensive.

- SECTR_AudioCue Cue = null

  The Cue to play from this source.

- bool Loop = true

  If the Cue should be forced to loop when playing.

- bool PlayOnStart = true

  Should the Cue auto-play when created.

Properties

- override bool IsPlaying [get]

  Returns true if the NoiseMaker is currently playing a sound.

Detailed Description

Playes a SECTR_AudioCue within a 3D volume.

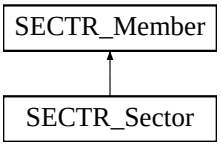
It's often desirable to represent a sound not as a single point, but as an entire region of space. RegionSources make that possible by efficiently computing the nearest point on the spline to the active listener, and positioning its sound instance at that location. This creates a very convincing illusing of the sound eminating from the entire spline, while using only one actual audio instance.

RegionSource supports any collider that Unity allows. However, for performance reasons it will default to using the AABB of whatever collider is used. If more accuracy is desired, raycasting can be enabled, which will determine the exact closest point (at some additional CPU cost.

2.46 SECTR_Sector Class Reference

Sectors represent discrete sections of the world, connected to one another by SECTR_Portal objects.

Inheritance diagram for SECTR_Sector:



Public Types

- enum BoundsUpdateModes { BoundsUpdateModes.Start, BoundsUpdateModes.Movement, BoundsUpdateModes.Always, BoundsUpdateModes.Static }

  Rules for how often to update the children and bounds.
- enum ChildCullModes { ChildCullModes.Default, ChildCullModes.Group, ChildCullModes.Individual }

  Modes for how to cull children (VIS only)

Public Member Functions

- void ConnectTerrainNeighbors ()

  Sets up the terrain neighbors structure.
- void DisonnectTerrainNeighbors ()

  Disconnects this terrain from all neighbors and vice versa. Works around a crash in some versions of Unity.
- void Register (SECTR_Portal portal)

  Informs the Sector that a Portal is connected into it. Should only be called by SECTR_Portal.
- void Deregister (SECTR_Portal portal)

  Informs the Sector that a Portal is no longer connected into it. Should only be called by SECTR_Portal.
- void Register (SECTR_Member member)

  Informs the Sector that a Member is in it. Should only be called by SECTR_Member.
- void Deregister (SECTR_Member member)

  Informs the Sector that a Member is no longer part in it. Should only be called by SECTR_Member.

- void ForceUpdate (bool updateChildren)

  Forces an update. Used only in special cases.
- void SectorDisabled (SECTR_Sector sector)

  Called when a Sector is being destroyed.
- delegate void MembershipChanged (List< SECTR_Sector > left, List< SECTR_Sector > joined)

  Delegate delcaration for anyone who wants to be notified when membership changes.

Static Public Member Functions

- static void GetContaining (ref List< SECTR_Sector > sectors, Vector3 position)

  Returns the list of Sectors that contain a given point. Sectors may overlap and are not exclusive, hence the list.
  Parameters

sectors	*List of sectors to write into.*
position	*The world space position for which to search.*

  Returns

  List of Sectors containing position.
- static void GetContaining (ref List< SECTR_Sector > sectors, Bounds bounds)

  Returns the list of Sectors that intersect an AABB. Sectors may overlap and are not exclusive, hence the list.
  Parameters

sectors	*List of sectors to write into.*
bounds	*The world space bounding box for which to search.*

  Returns

  The List of Sectors overlapping bounds

Public Attributes

- SECTR_Sector TopTerrain

  The terrain Sector attached on the top side of this Sector.
- SECTR_Sector BottomTerrain

  The terrain Sector attached on the bottom side of this Sector.
- SECTR_Sector LeftTerrain

  The terrain Sector attached on the left side of this Sector.
- SECTR_Sector RightTerrain

  The terrain Sector attached on the right side of this Sector.
- bool PortalDetermined = false

  Set to true if Sector membership should only change when crossing a portal.
- SECTR_Sector ForceStartSector = null

  If set, forces the initial Sector to be the specified Sector.
- BoundsUpdateModes BoundsUpdateMode = BoundsUpdateModes.Always

  Determines how often the bounds are recomputed. More frequent updates requires more CPU.
- float ExtraBounds = SECTR_Geometry.kBOUNDS_CHEAT

  Adds a buffer on bounding box to compensate for minor imprecisions.
- bool OverrideBounds = false

  Override computed bounds with the user specified bounds. Advanced users only.
- Bounds BoundsOverride

  User specified override bounds. Auto-populated with the current bounds when override is inactive.
- Light DirShadowCaster

  Optional shadow casting directional light to use in membership calculations. Bounds will be extruded away from light, if set.
- float DirShadowDistance = 100

Distance by which to extend the bounds away from the shadow casting light.

- ChildCullModes ChildCulling = ChildCullModes.Default

  Determines if this SectorCuller should cull individual children, or cull all children based on the aggregate bounds.

Properties

- static new List< SECTR_Sector > All `[get]`

  Returns a list of all enabled Sectors.
- bool Visited `[get, set]`

  Utility property for tracking during graph walks.
- List< SECTR_Portal > Portals `[get]`

  Returns all of the portals connected to this Sector.
- List< SECTR_Member > Members `[get]`

  Accessor for the members of this Sector.
- bool IsConnectedTerrain `[get]`

  Returns true if this Sector is setup as part of a grid of terrain tiles.
- bool CullEachChild `[get]`

  Returns true if each child should be culled individually.
- List< SECTR_Sector > Sectors `[get]`

  Returns all the Sectors that this object belongs to.
- List< Child > Children `[get]`

  Returns a flattened list of all relevant children.
- List< Child > Renderers `[get]`

  Returns the subset of the Children list that contains Renderers.
- bool ShadowCaster `[get]`

  Returns true if any child renderer components casts shadows.
- List< Child > ShadowCasters `[get]`

  Returns the subset of the Children list that cast shadows.
- List< Child > Lights `[get]`

  Returns the subset of the Children list that contains Lights.
- bool ShadowLight `[get]`

  Returns true if any child light components create shadows.
- List< Child > ShadowLights `[get]`

  Returns the subset of the Children list that create shadows.
- List< Child > Terrains `[get]`

  Returns the subset of the Children list that contains Lights.
- Bounds TotalBounds `[get]`

  Returns the union of the RenderBounds and LightBounds.
- Bounds RenderBounds `[get]`

  Returns the union of the Bounds of all child Renderers.
- bool HasRenderBounds `[get]`

  Returns true if the RenderBounds contains valid data.
- Bounds LightBounds `[get]`

  Returns the union of the Bounds of all child Lights.
- bool HasLightBounds `[get]`

  Returns true if the LightBounds contains valid data.
- bool Frozen `[get, set]`

  (Un)Freezes (i.e. disables updates and preserves bounds).
- SECTR_Member ChildProxy `[set]`

  Setting the ChildProxy will force this Member to use the children of another node as if they were its own. Only to be used in very specific circumstances.

- bool NeverJoin [set]

  Setting to true will prevent this Member from joining any sectors. Only to be used in very specific circumstances.
- bool IsSector [get]

  Returs if this Member is really a Sector.

Events

- MembershipChanged Changed

  Event handler for membership changed callbacks.

Detailed Description

Sectors represent discrete sections of the world, connected to one another by SECTR_Portal objects.

Sectors are roughly analagous to rooms in a building, with a unique shape, size and location. Objects that overlap the bounds of the Sector are considered to be contained in it, in the same way that a table or stove would be thought of as "in the kitchen". Sector bounds can overlap and membership is not exclusive; a SECTR_Member may be in multiple Sectors at once.

Like the rest of the system, Sectors are be completely dyanamic, and can transform and be enabled/disabled dynamically. Because the rooms in many games are completely static, marking a Sector as isStatic will enable some additional performance optimizations.

The size and shape of a Sector is defined by the union of the bounds of the Renderable Mesh children parented underneath it. Lights and other types of objects may be part of the Sector proper, but will not influence the "official" bounds.

As an implementation detail, Sector derives from SECTR_Member, primarily because every Sector needs the services that Member provides, and a little bit of special treatment besides.

Member Enumeration Documentation

enum SECTR_Member.BoundsUpdateModes [inherited]

Rules for how often to update the children and bounds.

Enumerator

Start Compute children on start. Update bounds on movement.

Movement Compute children and bounds on movement.

Always Compute children and bounds on every update.

Static Compute children and bounds on only on start.

enum SECTR_Member.ChildCullModes [inherited]

Modes for how to cull children (VIS only)

Enumerator

Default Cull Sector children individually, Member children as a group.

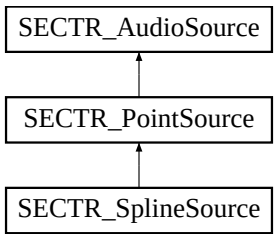
Group Cull children as a group.

Individual Cull each child individually.

2.47 SECTR_SplineSource Class Reference

Plays the specified SECTR_AudioCue at the nearest point along a spline to the listener.

Inheritance diagram for SECTR_SplineSource:



Public Member Functions

- override void Play ()

  Make some noise! Plays the Cue.
- override void Stop (bool stopImmediately)

  Stops the Source from playing.
  Parameters

stopImmediately	*Overrides any fade-out specified in the Cue*

Public Attributes

- List< Transform > SplinePoints = new List<Transform>()

  Array of scene objects to use as control points for the spline.
- bool Closed = false

  Determines if the spline is open or closed (i.e. a loop).
- SECTR_AudioCue Cue = null

  The Cue to play from this source.
- bool Loop = true

  If the Cue should be forced to loop when playing.
- bool PlayOnStart = true

  Should the Cue auto-play when created.

Properties

- override bool IsPlaying `[get]`

  Returns true if the NoiseMaker is currently playing a sound.

Detailed Description

Plays the specified SECTR_AudioCue at the nearest point along a spline to the listener.

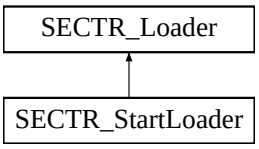
Many phenomena that emit sound (like streams or roads) are well described by splines. This component makes it easy to play sounds that mimic the behavior of these sources. The SplineSource will efficiently compute the nearest point on the spline to the active listener, and position its sound instance at that position. This creates a very convincing illusing of the sound eminating from the entire spline, while using only one actual audio instance.

Liberally adapted from http://wiki.unity3d.com/index.php?title=Spline_Controller

2.48 SECTR_StartLoader Class Reference

Loads SECTR_Chunk components that this object is in at Start and nothing more.

Inheritance diagram for SECTR_StartLoader:



Public Attributes

- bool FadeIn = false

  Set to true if the scene should start at black and fade in when loaded.
- float FadeTime = 2

  Amount of time to fade in.
- Color FadeColor = Color.black

  The color to fade the screen to on load.

Properties

- override bool Loaded `[get]`

  Returns true if all referenced Chunks are loaded. False, otherwise.

Detailed Description

Loads SECTR_Chunk components that this object is in at Start and nothing more.

StartLoader is designed to be combined with SECTR_LoadingDoor in order to make sure that the reference count of the initial Sector(s) work out correctly with the load/unload logic of the door.

StartLoader self-destructs immediately after doing its work in order to eliminate its overhead post-start.

2.49 SECTR_StartMusic Class Reference

Plays a piece of music on Start.

Inherits MonoBehaviour.

Public Attributes

- SECTR_AudioCue Cue

  The music to play on Start.

Detailed Description

Plays a piece of music on Start.

2.50 SECTR_StreamExport Class Reference

A set of static utility functions for exporting scenes and doing other stream related processing.

Static Public Member Functions

- static bool ImportFromChunk (SECTR_Sector sector)

  Re-adds the data from the specified Sector to the current scene. Safe to call from command line.
  Parameters

sector	*The Sector to import.*

  Returns

  Returns true if Sector was successfully imported, false otherwise.
- static bool ExportToChunk (SECTR_Sector sector)

  Exports the specific Sector into an external level file, deleting the current scene copy in the process. Safe to call from command line.
  Parameters

sector	*The Sector to export.*

  Returns

  Returns true if Sector was successfully exported, false otherwise.
- static void ExportSceneChunksUI ()

  Exports all of the Sectors in the scene, with user prompts and other helpful dialogs.
- static void ExportSceneChunks ()

  Exports all Sectors in the scene. Safe to call from the command line.
- static void ImportSceneChunksUI ()

  Imports all of the Sectors in the scene, with user prompts and other helpful dialogs.
- static void ImportSceneChunks ()

  Imports all exported Sectors into the scene. Safe to call from the command line.
- static void RevertSceneChunksUI ()

  Reverts all of the imported Sectors in the scene, with user prompts and other helpful dialogs.
- static void RevertSceneChunks ()

  Reverts all imported Sectors into the scene. Safe to call from the command line.
- static void WriteGraphDot ()

  Writes out the current scene's Sector/Portal graph as a .dot file which can be visualized in programs like GraphVis and the like.

Detailed Description

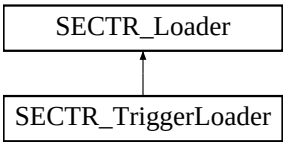
A set of static utility functions for exporting scenes and doing other stream related processing.

In order to stream a scene, we need to split the base scene up into multiple levels. We use levels and additive addition instead of Resource Bundles because they take less memory during load and do not cause assets to be duplicated on disk.

2.51 SECTR_TriggerLoader Class Reference

(Un)loads a list of SECTR_Chunk objects based on Unity Trigger events.

Inheritance diagram for SECTR_TriggerLoader:



Public Attributes

- List< SECTR_Sector > Sectors = new List<SECTR_Sector>()

  List of Sectors to load when entering this trigger.
- bool UnloadOnExit = true

  Should the Sectors be unloaded when trigger is exited.

Properties

- override bool Loaded `[get]`

  Returns true if all referenced Chunks are loaded. False, otherwise.

Detailed Description

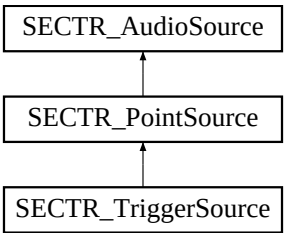
(Un)loads a list of SECTR_Chunk objects based on Unity Trigger events.

The component allows developers to load a list of Sectors when the player enters a particluar region of space. TriggerLoader uses standard Unity trigger events, so any Collider can be used, provided its marked as a trigger.

2.52 SECTR_TriggerSource Class Reference

Playes a SECTR_AudioCue when a trigger is activated.

Inheritance diagram for SECTR_TriggerSource:



Public Member Functions

- override void Play ()

  Make some noise! Plays the Cue.
- override void Stop (bool stopImmediately)

  Stops the Source from playing.
  Parameters

stopImmediately	*Overrides any fade-out specified in the Cue*

Public Attributes

- SECTR_AudioCue Cue = null

  The Cue to play from this source.
- bool Loop = true

  If the Cue should be forced to loop when playing.
- bool PlayOnStart = true

  Should the Cue auto-play when created.

Properties

- override bool IsPlaying `[get]`

  Returns true if the NoiseMaker is currently playing a sound.

Detailed Description

Playes a SECTR_AudioCue when a trigger is activated.

TriggerSource supports any collider that Unity allows, provided it's marked to be a trigger.

2.53 SECTR_Wanderer Class Reference

A component that will wander the scene by pathing through the Sector/Portal graph.

Inherits MonoBehaviour.

Public Attributes

- float MovementSpeed = 1

  The speed at which the wanderer moves throughout the world.

Detailed Description

A component that will wander the scene by pathing through the Sector/Portal graph.

Wanderer simply picks a goal sector, plots a path to it, and then follows that path, going through the center of each Portal and Sector along the way. Useful for testing and demoing objects moving through the world.

Questions or Problems

support@makecodenow.com