VVUnity
The VVAudio Unity 3D Audio System

The VVAudio Unity 3D Audio System includes a demo project containing a set of scripts and libraries that are meant for a developer to use in their own projects.

VVUnityDemo

The VVUnityDemo project contains a Google Cardboard application designed to demonstrate VVAudio's Unity 3D Audio System. There are four simple objects which represent four audio sources playing a string quartet (Bach, Preludes and Fugues, Book 1, Fugue 24). After a few seconds they begin to move, illustrating the smooth changes localization as the distance and angle change.

The demo GUI has controls for overall rendering as well as the details of the synthetic binaural process. There is also a control which can make the objects move silly fast, which demonstrates fractional sample, distance-based delays naturally creating doppler effects.

Scripts

In order to use VVAudio's Unity Audio System, you must copy the following scripts, as well as VVDSP.dll to your project Assets folder. You may optionally copy the appropriate VVUnityPlugins packages into the Assets/Plugins folder.

VVListener

There should be a single instance of the VVListener script in a scene, usually attached to the main camera. This script is the central controller for the others. In addition to doing the ambisonic rotation and decoding and bringing the ambisonic busses back into the Unity audio chain, VVListener also sets the parameters for all the spatializers in a scene.

Native libraries are available on some platforms as described below. Use native will be off by default if they are not available.

Use Native Libraries?	True	use native c++ if available
	False	use script regardless

Render type controls the conversion of the final ambisonic mix to stereo.

Render Type	XY	A simple, non-binaural ambisonics decoder

	Synth5	Linear ambisonic decoder combined with synthetic HRTFs (script only)
	Synth16	Parametric ambisonics decoder combined with synthetic HRTFs (16 HRTFs, native only)
	Listen1025-16	Parametric ambisonics decoder combined with measured HRTFs (16 HRTF's, native only)
	Listen1025-4	Parametric ambisonics decoder combined with measured HRTFs (16 HRTF's, native only)

Spatialize Mode overrides how the VVSpatializer scripts encode signals. An all ambisonic mix can be useful to limit CPU use or to ensure speaker compatibility.

Spatializer Mode	Binaural	A mix of binaural and ambisonics, based on distance
	Ambisonic	Ambisonics only, regardless of distance

The following parameters control the room response.

Room Width, Length & Height	Control the early echos in different directions
Reverb Size	Controls the length of the reverb tail, following the early reflections
Reverb Gain	Sets the gain of the room response return to the main b-format bus

Forward emphasis is a central control for all spatializers. It is applied to both binaural and ambisonic signals. This control can be used to help disambiguate front and back sounds.

Frontal Emphasis	Controls the degree to which rear sounds are attenuated

The next set of parameters control the synthetic binaural HRTFs used at several places in the signal chain. These controls can be used to customize the binaural experience for a given user.

Head Size	Controls the delay between left and right
Ear Size	Controls the frequency of pinnae effects
Torso Size	Controls the frequency of low frequency effects

VVSpatializer

The VVSpatializer script should be attached to an Audio Source. It will intercept the Unity audio chain and send the result to either the ambisonic or binaural buss. The spatializer implements fractional sample delay, linear or log distance falloff, forward emphasis, and a combination of binaural and ambisonic panning.

Is Object?	True	Binaural and/or ambisonics, depending on distance (see d1-d4)
	False	Ambisonic panning
Object Number	Each audio object should be given a different object number	

Each VVSpatializer has four distance parameters that control rendering as follows:

Distance	< d1	Binaural
	d1 to d2	Fade from binaural to ambisonic
	d2 to d3	Ambisonic
	d3 to d4	Fade out
	> d4	Off, minimal CPU use

Note that the following should be true: d1<d2<d3<d4. To make an object always binaural set d1 (and therefore d2-4) greater than any distance used. (Currently limited to 20 meters)

Note that, since ambisonics output requires four channels and Unity only passes two, only the binaural outputs are available to send to a mixer group. That part of a source that goes to ambisonics only shows back up at the listener.

VVAmbiClip

The VVAmbiclip script should be attached to an Audio Source containing a four channel B-format clip. Since Unity doesn't send all four channels down the audio chain, this script reads the clip data directly and sends it to the ambisonics bus for later rotation and rendering by the listener. Since it run only when the audio source it is connected to runs, playback can be controlled normally at the audio source.

Ambisonic clips can optionally send to the second B-format bus for static , non-rotated sounds. An example might be background music that should not be rotated with head motion.

Is Static?	True	Sent to second B-format bus which is not rotated
	False	Sent to first B-format bus which gets rotated with head movement

VVScaler
VVScaler is a small script that intercepts the audio stream just to measure its volume then scales the object it is attached to from .5 to 1.5 times it original size from -60 to 0 dB. Note that this script reads the scale once at startup so it will not interact well with other scripts that change the scale.

Busses
There are three internal busses used by the VVUnity scripts:

B-format Bus 1
This is the main ambisonic bus that is rotated by head movement and decoded.

B-format Bus 2
This bus does not get rotated before decoding. It is meant for background music or other sound elements that should stay static with relation to the listener.

B-format Reverb Bus
An ambisonic mix that is fed into the room response unit. The results of this are then mixed with the other B-format busses before rendering. The room response is rotated with the listener.

Native vs. Script
All functionality is available in pure script for and thus should be compatible with all platforms. VVUnityDSP.dll is a cross platform C# library for this purpose. There are currently native libraries for OSX and Windows called VVUnityPlugins.bundle and VVUnityPlugins.dll respectively. The native libraries offer better performance and more sophisticated ambisonic to binaural decoding.

Performance
The use of CPU can be carefully managed.   Most important is the use of native libraries as described for VVListener above. OSX and Windows are available now. Android and iOS versions will be developed, of course.

Next, the binaural rendering of objects takes much more CPU than simple ambisonic panning. This is controlled primarily by setting an audio source to be an object and thus potentially binaurally rendered, or not an object and thus always ambisonically panned. Then, for those sources that are objects, there is a distance formula that will further manage CPU use. After objects are faded to ambisonics, past d2, the CPU use will be much lower, then after d4 it will stop processing completely.

Last in the signal chain, but potentially a CPU hog, is ambisonic to binaural decoding. The XY decoder is very fast and good enough for some purposes. The SynthSquare decoder (only in script right now) is the next most CPU intensive, but gives some binaural rendering of the b-format busses. The native Synth16 and Listen1025-16 decodes are by far the most intensive. The Listen1025-4 is somewhere in between.

Speaker Compatibility

The XY ambisonics decoder is not binaural and thus is compatible with speaker (instead of headphone) output. To be completely speaker compatible, all spatializers should be switched to ambisonics as well using the control on VVListener.