ECHOLOCATION LTD

3D SOUND FOR 2D GAMES - USER GUIDE

INTRODUCTION

This 3D sound engine plugin for Unity will transform your 2D games and give you the edge in user engagement and experience. Developed after over 25 years research into bat and human echolocation; it taps into the brains capability of positioning sound in 3D space.

This software will give you the full three dimensions of sound; all you have to provide is the sound clip and game object. The sound engine does all the work by tracking the game object; and positions the sound with respect to the object's position in 2D World or Screen coordinates. There's also an ability to position the object off-screen by providing sound dimension sizes and offsets.

But the key feature of this plugin is its ability to play multiple sound sources in 3D without overloading the CPU. It can comfortably play around 50 3D sounds simultaneously on a Smartphone (many more on a PC), which would be useful in an interactive and highly charged part of a game. This can be seen in the modified demo version of Unity's Space Shooter game, which can be found via the video link on the Unity Asset Store.

THE PLUGIN INSTALLATION

The effect plugin will be available for download via the Asset Store. It will appear as two DLLs; one for the editor interface, and the other for the sound engine itself. There will also be a bouncing ball demo application that demonstrates the sound engine in action.

Once these have been successfully imported into your game or application, you can add the '3D Sound For 2D Games' component via two menu options. First select the game object in the Hierarchy section of Unity you wish to attach the component to. Then either select from the top menu 'Component->Echolocation Ltd->3D Sound For 2D Games', as per Figure 1:

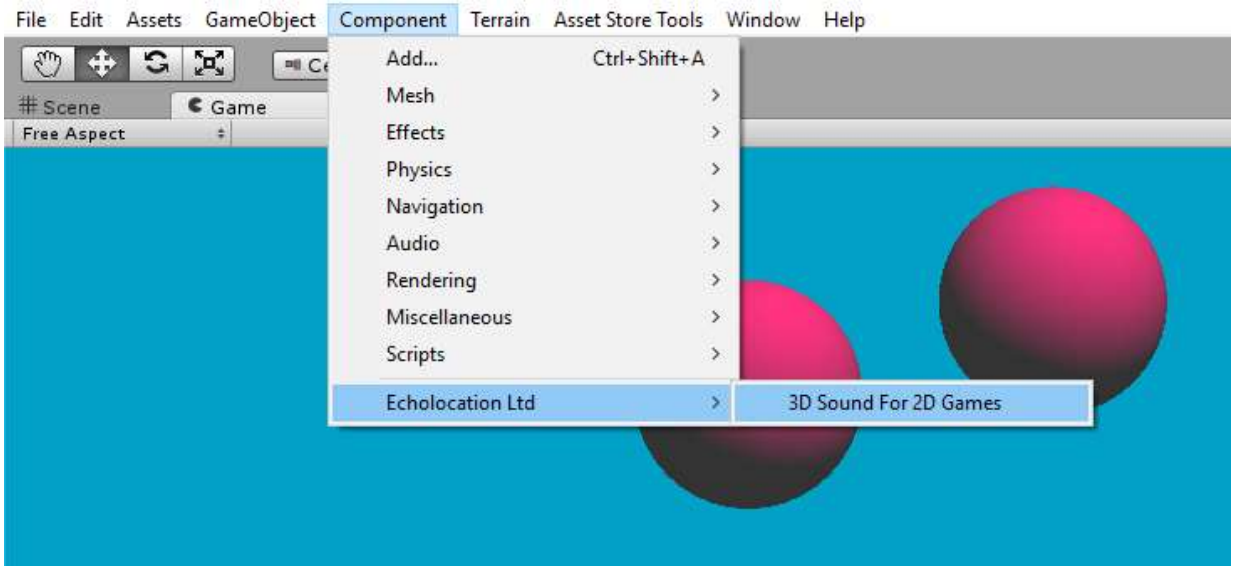


Figure 1

Or select from the 'Add Component' button in the Inspector and navigate to 'Echolocation Ltd->3D Sound For 2D Games', as per Figure 2.

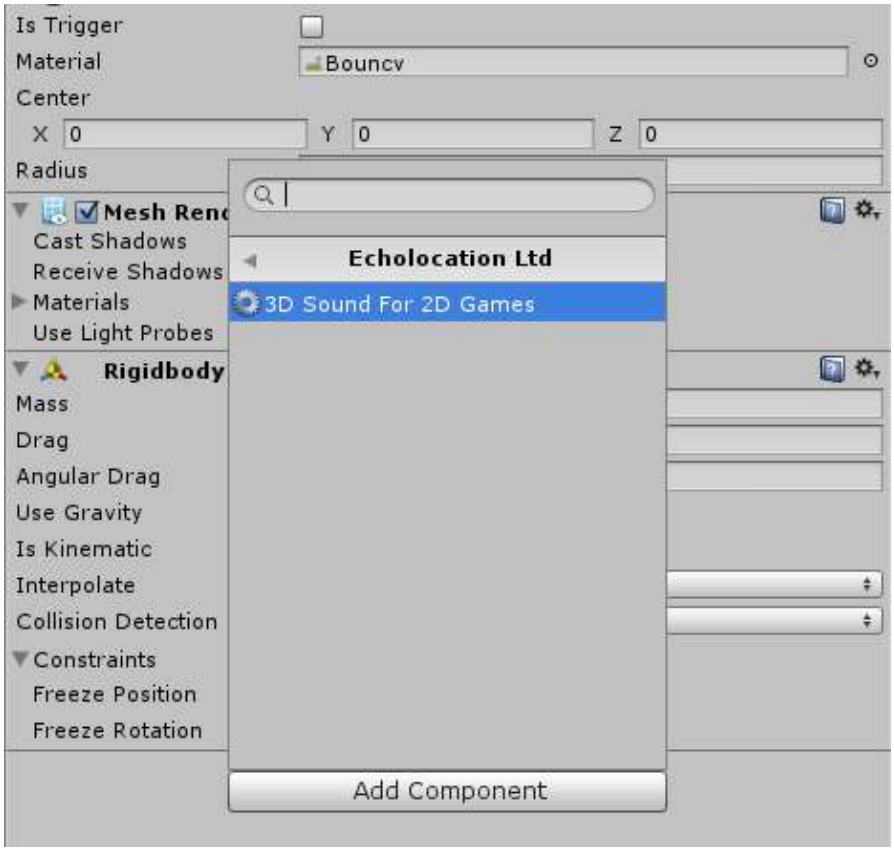


Figure 2

THE PLUGIN EDITOR EXTENSION

Figure 3 shows a screenshot of the plugin after it's been installed and attached to a game object.

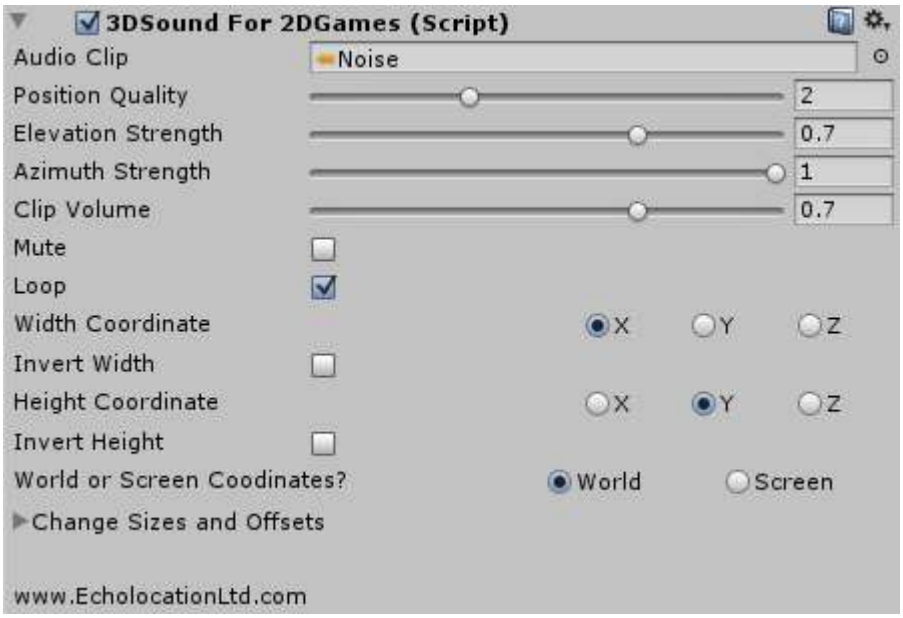
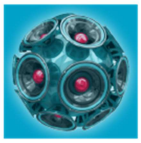


Figure 3



The first option available is to provide a link to your chosen sound clip. Note that this clip must be stereo, and sampled at 44.1 kHz or above. The latter is necessary because the sound engine requires a wide bandwidth in order for it to function well, so the richer the sound is in harmonics, the better will be the positioning.

The next option is the Position Quality, which is selectable between 1 through 4 in integer steps. The higher the value, the smoother will be the transition in sound positioning, but at the expense of CPU time. The optimal quality level is 2, which is the default; but if you have a wider, off-screen sound movement you may want to increase this.

The Elevation Strength is provided so you can fine-tune the sounds positioning strength in the vertical plane. A setting of '0' will provide a very flat sound, but increasing it will add the height component. You won't need this setting to be very strong over headphones, but may need to increase it if you are listening over loudspeakers.

The Azimuth Strength can be used to alter the strength of the width of the sound. A setting of '0' gives zero panning, but increasing it increases the panning up to a maximum at '1'. This is useful if you want to fine-tune the horizontal width of the sound engine, or if you want the panning effect to work effectively over loudspeakers.

The Clip Volume control does what is says, but note that it is the overall volume controller for the clip, i.e. it controls the volume level for all attached game objects. You can control the sound level for each individual game object, but this is only achieved via a script.

Mute simply mutes the sound and Loop allows the clip to be looped or not.

The next few controls are a bit trickier. The first set of radio buttons allows the user to link the sound's horizontal or width effect with a game object's x, y, or z coordinates. There's also a tick box to invert the polarity of the width coordinate.

The second set does the same for the vertical or height effect, and there too is a tick box for its polarity. The default is as mapped out per 2D screen space with width in the x direction and height in the y direction. However, if your games' 2D coordinate system is different, you can easily remap them by changing the radio selections appropriately.

The last radio selection allows the user to position the sound in 3D space according to the games object's position in world coordinates, or according to the position of the game object with respect to the screen. The World selection means the screen dimensions are ignored and the sound engine will not try to change its 3D space to fit it. The Screen selection means the screen dimensions will not be ignored, and will be useful for games or applications that change their graphics to fit the screen size.

The final sets of options are available via a fold-out extension as per Figure 4. Note that you need to click on the arrow next to the 'Change Sizes and Offsets' label to expose these settings.

If the World Coordinates radio button is selected, the top section is enabled, whereas if the Screen option is selected, the bottom section is enabled respectively.

These settings allow the user to change the width and height dimensions or sizes of the sound, and offset the centre from the world origin or screen centre. These settings are useful if the game or application needs to position the sound off-screen, or wants to restrict the sound dimensions to the top half of the screen. For example, the Submarine game demo demonstrates the former; with a torpedo being heard on the right sometime before it appears on screen. It then moves across the screen and fades away as it transfers off the screen to the left.

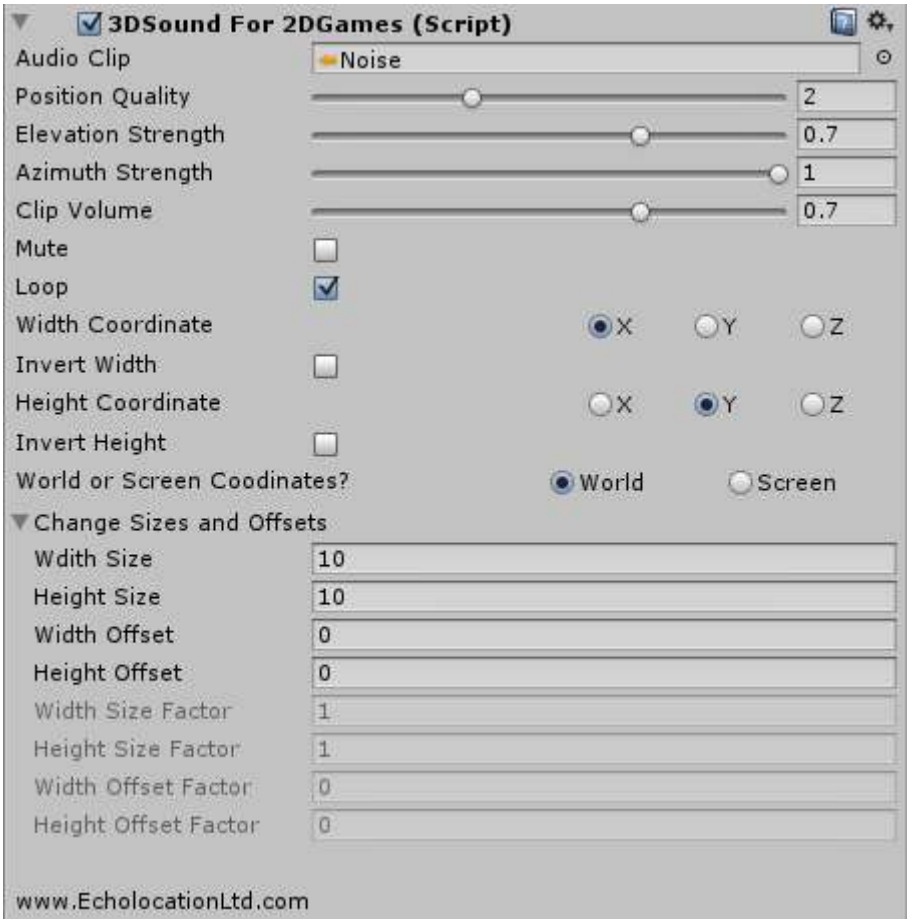


Figure 4

If you want the sound to stay within +/-5 units in world coordinates, and there is zero offset from the origin, you don't need to change the top settings. To narrow or widen the width and/or height of the sound, you need to change these settings appropriately. Equally, if you want to offset the width or height settings, you need to change the Offset settings away from the origin. The default is the origin.

This is also true for the Screen settings; if you want to stay within the screen dimensions with zero offset from the centre, you don't need to change anything.

One final comment about the Editor Extension; when you hit the play button, unless a sound source object has not already been added, the plugin will immediately create a sound source below the extension.

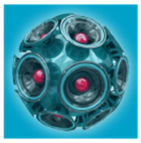
CONTROLLING THE PLUGIN VIA C# SCRIPT

The plugin currently only works with C# scripting.

The accompanying bouncing ball demo contains a small script that is used to pass information to the sound engine so it can calculate where the various sounds needs to be positioned in 3D space with respect to the game object's position.

The only function that you will need to use is the following, with its overloaded counterpart:

Play3D(GameObject go)
Play3D(GameObject go, float volume)



Before calling these methods, you need to find or get a link to the game object you want to position in 3D sound space. Once this has been collected, you need to get a copy of the sound engine component. From here you can then call the Play3D function, passing it the game object component and optionally the volume of the sound for the game object (default is full volume).

Note that the sound engine will keep a record of the game object, and frequent calls to Play3D will be required to update the game object's position information. Also, once the game object has been destroyed, the sound engine will forget about it, and will stop playing the sound linked with this object.

Also note that in addition to the plugin keeping track of the game object; it also keeps track of the playback position of the clip for the various game objects. So if one game object calls the method at time = 0 and the clip immediately starts playing; another game object can call the same method at time = 1 second, and the second version of the clip will start playing in a different 3D position 1 second later. The sound outputs (read from the same clip) will thus be out of synch; improving the realness and richness of the sound. This is demonstrated in the Space Shooter demo video.

THE BOUNCING BALL DEMO

The plugin is accompanied by a demo that contains three bouncing balls bounded by walls, a ceiling and an irregular floor. When you hit the play button, they fall from the top of the screen and bounce around, off these boundaries until they come to rest. The user can pick up any ball using their mouse, move it around the screen and then let it go again to continue bouncing around. The user will be able to hear the sound positions change and move with respect to the position of the ball(s). The best effect can be heard over headphones, but the effect still works well over loudspeakers.

The sound engine plugin keeps track of the objects through constant calls to the Play3D function via the Update() method in the AudioControl script. When the play button is pressed, the plugin creates its own sound source, and starts to play the sound specified in the editor; positioning this sound with respect to the position of the game object in either world or screen space.

Note that only Balls 1 and 2 play continuously, but you can hear sound for Ball 3 by clicking anywhere on the screen other than a ball. If you hold the left mouse button down, it will repeat the sound and the sound will follow the ball until you lift your finger off the mouse button or another ball gets in the way. This has been included simply to demonstrate un-looped calls to the method.

USING THE PLUGIN IN YOUR GAME

You could have an editor window for each game object and control them separately, but this would not be very efficient or helpful. One of the major benefits of using this sound engine is that you can track multiple game objects and only have a single instance of the plugin. You are of course restricted to only one sound clip for this, but if you have multiple cloned objects with the same sound appearing repeatedly in the game (such as laser bolts in the Space Shooter game demo), they will each be tracked and the corresponding sound position given. This provides a huge array of sounds that will enliven any dull game up. All you need to do to get this working with the game object Instantiate feature in Unity is to keep track of the cloned game objects, frequently passing the game object to the sound engine plugin via the Play3D methods.

Another benefit to this engine is the scriptable volume control. This can be used to fade objects in and out of the action, or anything that requires tighter control over individual sounds. Note that this scriptable volume control is not directly linked to the one in the editor, as the editor one controls the overall volume of the clip, and not the sound for each individual game object.