gameCODA™

All platforms

Concepts Guide

Issue 2.0

Sensaura

CREATIVE®

CODA/005/0804/1

Copyright

Trademarks

Disclaimer

Document number: CODA/005/0804/1

Publication date: 29 July 2004

Sensaura
3D Labs Building
Meadlake Place
Thorpe Lea Road
EGHAM TW20 8HE
UNITED KINGDOM

Telephone: +44 1784 476 777
Facsimile: +44 1784 476 770
http://www.sensaura.com/
http://www.gamecoda.com/

Table of contents

1 Introduction

GameCODA incorporates a number of sound concepts that may not be familiar to all developers. It is essential to have a clear understanding of these in order to make effective use of GameCODA as the audio engine for a game. This document explains these concepts and introduces several advanced Sensaura algorithms. A number of companion White Papers cover some of these topics in much greater detail for the interested reader.

The concepts discussed in this document are:

- Buffer and source basics;
- 3D sound basics;
- Sources and the listener;
- Speaker and headphone options;
- Distance rolloff;
- Source directivity;
- Doppler;
- MacroFX™;
- ZoomFX™;
- EnvironmentFX™.

Although basic GameCODA can be very effective in a game, the use of the more advanced features can greatly add to the impact of 3D audio if used appropriately. Doppler effects can enhance the sense of fast movement, especially when the sound source is close to the listener. MacroFX heightens the realism of very close sound sources and ZoomFX makes large game objects actually sound large too. EnvironmentFX provides an immersive feeling that makes you really believe that you're in the game environment.

2 Buffer and source basics

GameCODA differentiates between buffers and sources[1]. A buffer describes a block of raw audio data and a source is an abstract object in 2D or 3D space which emits sound by playing a buffer or buffers.

2.1 Buffers

A buffer describes a block of audio data. Its properties include its size, the format of the data, and optional loop points. The loop points of a buffer determine the point at which the audio should be looped when played.

2.2 Sources

GameCODA supports both 2D and 3D sources. 2D sources are positioned on a 2D plane and can be panned from left to right, and vice-versa. 3D sources can be positioned at any point in 3D space. Sources have attached to them a buffer or queue of buffers. When the source is played the buffer is processed and sent to the speakers, as required. GameCODA supports many forms of processing from basic pitch and level changes to 3D positioned reverberant sources.

[1] This can be contrasted with Microsoft DirectSound API where no distinction is made between a buffer and a source.

3 3D sound basics

Using only two ears, a human can determine the direction of a sound source in 3D space. The acoustic waves that arrive at each ear cause movement of the eardrums, which in turn can be represented as a stereo audio signal that incorporates the 3D information. It follows that, in theory, it should be possible to create the same effect synthetically, via headphones. GameCODA incorporates the technology to do just that and more.

Consider a single point sound source. The major cues that the brain uses to determine its direction in 3D space are:

- inter-aural time delay (ITD) – the time difference between the arrival of the wavefront at the closer ear ("near ear") to the sound source and its arrival at the other ear ("far ear");
- inter-aural amplitude difference (IAD) – the difference in loudness between the sound arriving at the near ear and that arriving at the far ear;
- pinna effects – the shaping of the frequency spectrum caused by the non-symmetrical shape of the outer ear, or pinna.

The white paper "An Introduction to Sound and Hearing" describes these cues in more detail and includes a wealth of background information on 3D sound perception.

GameCODA creates 3D effects by carefully mimicking these cues for each individual sound source, so you are subjected to the same acoustic signals through headphones that you would be subjected to directly in real life. Since your ears hear exactly the same in each scenario, your brain interprets the synthesised sounds as coming from real locations in 3D space.

GameCODA is not limited to headphone listening. The additional complications presented by speaker listening are discussed in Speaker and Headphone Options.

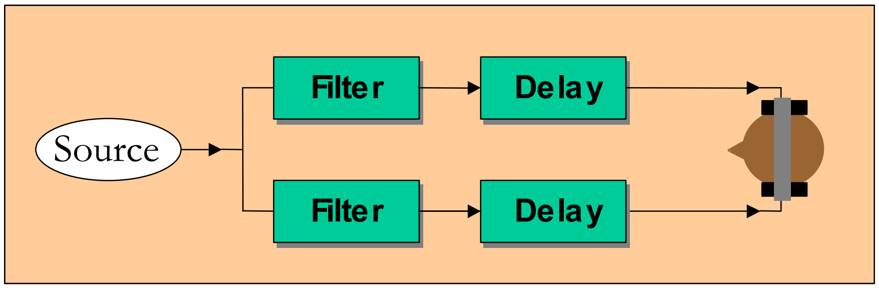

Figure 1: Basic 3D synthesis using filters and delay

Basic synthesis of these cues for a sound source is not complex, as can be seen in Figure 1. By updating the filter characteristic and delays frequently, the sound source can be made to appear to move smoothly through 3D space.

Although these basic principles are simple to implement, there are two design aspects that require expertise to achieve a good 3D synthesis solution. Firstly, the accuracy of the synthesis is almost totally dependent on the set of filter characteristics used and, secondly, the implementation of the filtering must be very efficient in order to avoid excessive processor usage. Sensaura's GameCODA middleware solution achieves the former goal through years of research, resulting in Sensaura's Digital Ear technology and achieves the latter by heavily optimised code designed to make maximum use of the available hardware on each target platform.

Digital Ear technology results in a table of over 1000 pairs of filter characteristics with associated time delays, each representing one unique point in 3D space. More information about the design of these filters can be found in the white paper "Digital Ear technology". Each pair of filters is called a Head-Related Transfer Function (HRTF).

Readers will be familiar with the term "render" as applied to graphics processing. We also use this term to describe the audio signal processing operations (i.e. HRTF filtering) that must be performed on a monophonic signal to position it in 3D space.

3.1 Sources and the Listener

In order to select the correct Digital Ear filters at any time, it is only necessary to know the relative positions of the sound source and listener, and the orientation of the listener. The absolute positions of the source and listener are unimportant. The orientation of the source is also unimportant if it is omni-directional. However, the GameCODA API allows the developer to specify the absolute positions of sources and the listener, and the listener orientation, as this is more convenient and fits well with graphics engine APIs. A left-handed or right-handed Cartesian co-ordinate system may be used.

GameCODA allows multiple sources to be created with each representing a separate sound source. Usually, the same co-ordinates will be used for the graphical position and audio position of each object. Normally there is only one listener, whose position and orientation should be set to represent the "listening point" in the game environment, just as one has a viewpoint represented by camera position and orientation in many games. Often, the developer will set the listener parameters to correspond to the position and orientation of the player or camera. Some experimentation may be required to obtain the best results, as the best camera view may not be the best "audio view", and indeed the two need not correspond exactly.

The GameCODA API supports multiple listeners, where each sound source is rendered for each listener simultaneously. The provision of multiple listeners is primarily to support split screen modes that are popular in some console games. Unfortunately, the hardware limitations of all current consoles mean that each player cannot get their own audio feed – i.e. the rendered audio for all listeners is mixed together. This is definitely non-ideal but the provision of multiple listeners has been one of the top requested items from developers.

A single vector cannot fully describe the listener's orientation, so two vectors are used, one emerging from the top of the listener's head (the "top vector") and the other from the front of his face (the "front vector").

3.2 Under the hood

The GameCODA API operates with source positions, position of listener(s) and listener orientation vectors. Internally, GameCODA calculates just two angular parameters (azimuth and elevation) from the source position, listener position and listener orientation, in order to look up the correct Digital Ear filter pair in the Sensaura HRTF array, as shown in Figure 2.

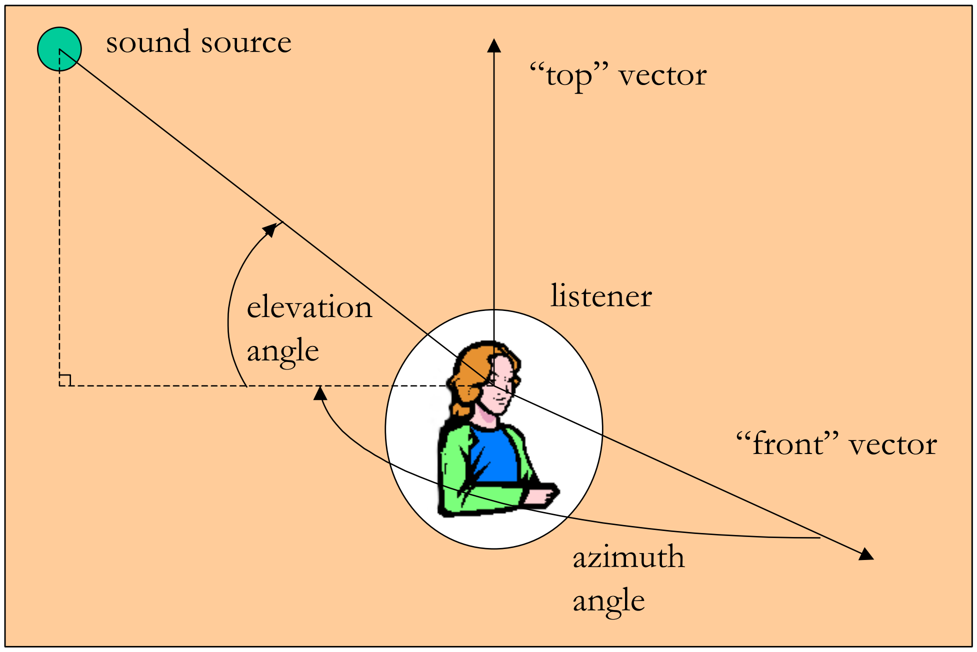

Figure 2: Azimuth and elevation uniquely specify source direction

If the "horizontal plane" is defined as the plane passing through both of the listener's ears such that the front vector lies in the plane (and the top vector is perpendicular to the plane), the azimuth and elevation angles are defined as follows:

Azimuth is the angle in the horizontal plane between the front vector and a perpendicular dropped from the sound source to the horizontal plane. 0° azimuth is directly ahead, 90° is to the right, 180° is behind and -90° is to the left of the listener.

Elevation is the angle that the sound source is elevated above the horizontal plane. 0° elevation is in the horizontal plane, 90° is directly above the listener and -90° is directly below.

4 Speaker and headphone options

So far, our discussion has focussed on headphone listening for ease of understanding, but in fact GameCODA is equally effective using a pair of speakers for output (or, if the hardware supports it, 4, 5.1, or other multi-channel configurations). However, simply feeding the speakers with the same signals (i.e. outputs from left and right HRTF filters) as the headphones does not work, the reason being that each of your ears hears not only the intended signal, but also the unwanted "crosstalk" from the other speaker, as shown in Figure 3. This crosstalk interferes with the filtering that we have carefully designed, resulting in a less effective positioning of 3D sounds.

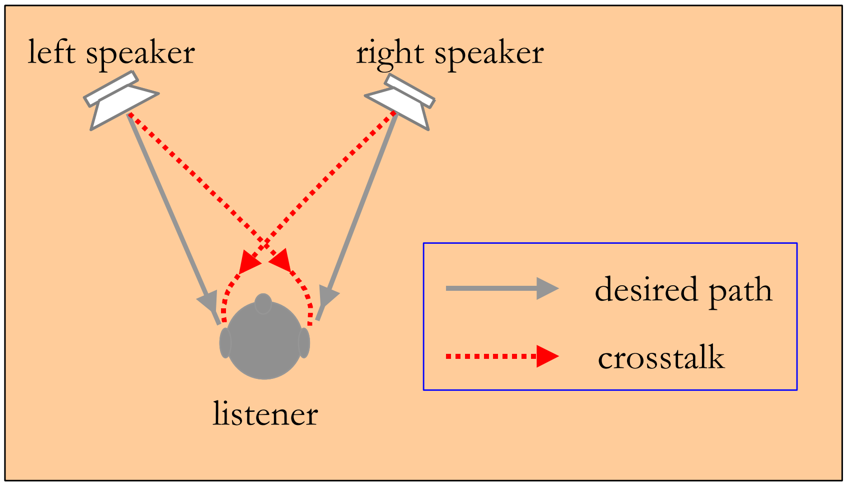

Figure 3: Desired signals and unwanted crosstalk

Fortunately, there is a solution, called "crosstalk cancellation", that removes the unwanted crosstalk and restores the full spatial experience, impossible as this may seem. The white paper "Transaural acoustic crosstalk cancellation" covers this area of the technology in some depth, and the paper "XTC crosstalk cancellation" contains a description of further enhancements, for example the optimisation of crosstalk cancellation for different speaker geometries. In fact, because the acoustic effect of using headphones is subtly different to listening on speakers, Sensaura GameCODA also uses an entirely different[2] table of filter characteristics for each output method for the very best end result.

[2] Due to memory constraints, on some platforms GameCODA uses common HRTF filters for both headphone and speaker configurations.

The developer should always provide a facility to allow the player to specify the output mode, be it headphones or one of the speaker configuration options, and communicate this choice to GameCODA to allow it to select the appropriate algorithms. For speaker configurations, GameCODA allows the separation of the speakers with respect to the listener to be specified. For example, narrowly separated stereo loudspeakers (e.g. built into a TV) or widely separated stereo loudspeakers (e.g. external hi-fi speakers).

5 Distance rolloff

The starting point for the GameCODA distance model is a *reference distance* (R_{REF}) that must be specified for each source (or left at the default of one distance unit[3]). This is the distance at which the rolloff gain is unity (i.e. the source is neither attenuated or amplified).

To calculate the amplitude gain (G_D) at any given distance (R_D), the ratio of the reference distance to this distance is raised to the power of a global setting – the *rolloff* (*ROLLOFF*).

$$G_D = \left(\frac{R_{REF}}{R_D} \right)^{ROLLOFF} \tag{1}$$

This definition is both simple and flexible. At the default rolloff setting of 1.0, behaviour is exactly the same as DirectSound3D with a rolloff factor of 1.0. Increasing or reducing the rolloff factor has a corresponding effect on the resultant gain rolloff effect. Here, behaviour differs from DirectSound3D (where its rolloff factor scales the difference between the distance and the reference point – an operation that has no real world analogy).

Why do we need different reference distances for different sources? The answer to this is due to the common practice of normalizing the gain of stored PCM sound samples (i.e. peaks extending to +32767/-32768 for a sample stored in 16-bit format). Without any adjustment of overall source gain or of the reference distance, a sound sample of an insect would sound as loud as that of a train engine (for example) at a given distance – clearly not a real-world situation.

The simple gain calculation is modified for distances less than a specified minimum and gain is clamped to what it would be at the minimum distance:

$$G_{MIN} = \left(\frac{R_{REF}}{R_{MIN}} \right)^{ROLLOFF} \tag{2}$$

[3] With the exception of the MacroFX radius (which relates directly to the size of a human head), the calculated gain due to distance is independent of the measurement units. However, for the examples given, the reader may find it convenient to think of distances in metres.

Similarly, for distances greater than a specified maximum the gain is clamped to what it would be at the maximum distance, or it is muted altogether (depending on a flag specified when the 3D source was created):

$$G_{MAX} = \left(\frac{R_{REF}}{R_{MAX}} \right)^{ROLLOFF} , G_{MAX} = 0 \qquad (3)$$

By way of comparison, DirectSound3D's minimum distance combines the functions of GameCODA's minimum and reference distances. Behaviour at maximum distance is the same. For an easy migration from DirectSound to GameCODA, simply replace each call to set the minimum distance by a pair of calls to set both the minimum and the reference distance (to the same value).

If MacroFX is enabled for a source, for distances less than one metre the MacroFX algorithm overrides the above calculations (and minimum distance clamping, if less than one metre, is ignored). See MacroFX for details.

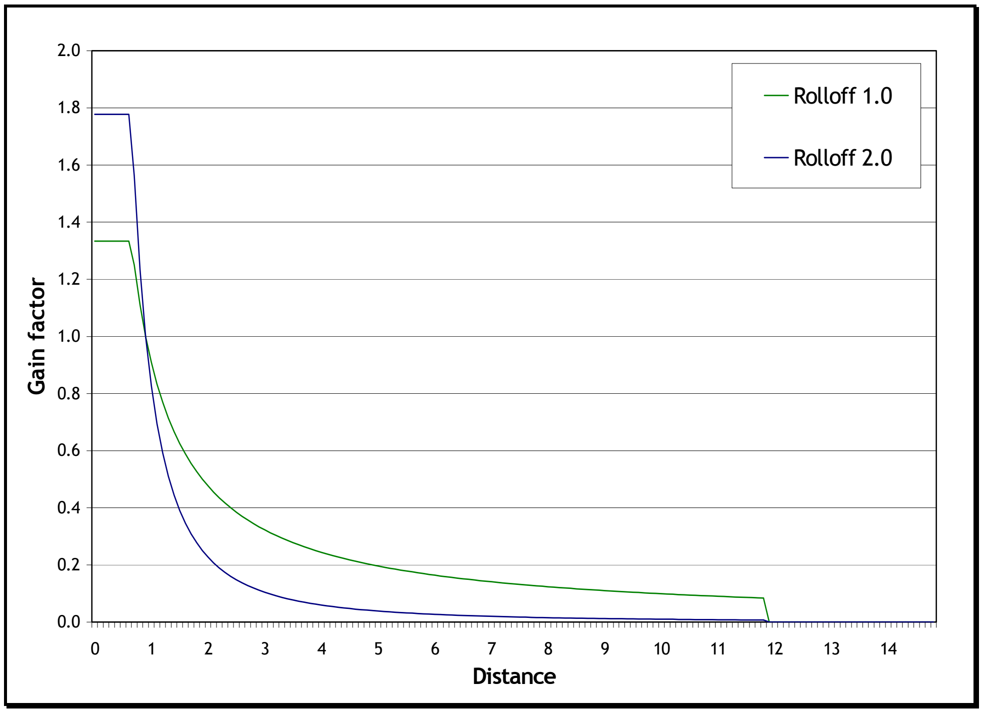

Figure 4: Effect of different rolloff values

Figure 4 shows the effect of choosing different rolloffs. In this example, minimum distance is 0.75 units, reference distance is 1 unit, maximum distance is 12 units and the mute at maximum flag is set. The only difference between the two curves is the choice of rolloff: 1.0 or 2.0. Notice that with a rolloff of 2.0 the curve is much steeper and beyond the reference distance the gain tends towards a lower value. Note also that both plots show a gain of 1.0 at 1 distance unit (as we would expect since they have the same reference distance).

Since GameCODA's rolloff is a global setting, in practice we cannot do what Figure 4 might suggest – have sources with different rolloff values. What we can do though is set different reference distances. An example of this is shown in Figure 5. The rolloff for each plot is 1.0, the reference distance is either 1 unit or 5 units and all other parameters are as in the previous example.

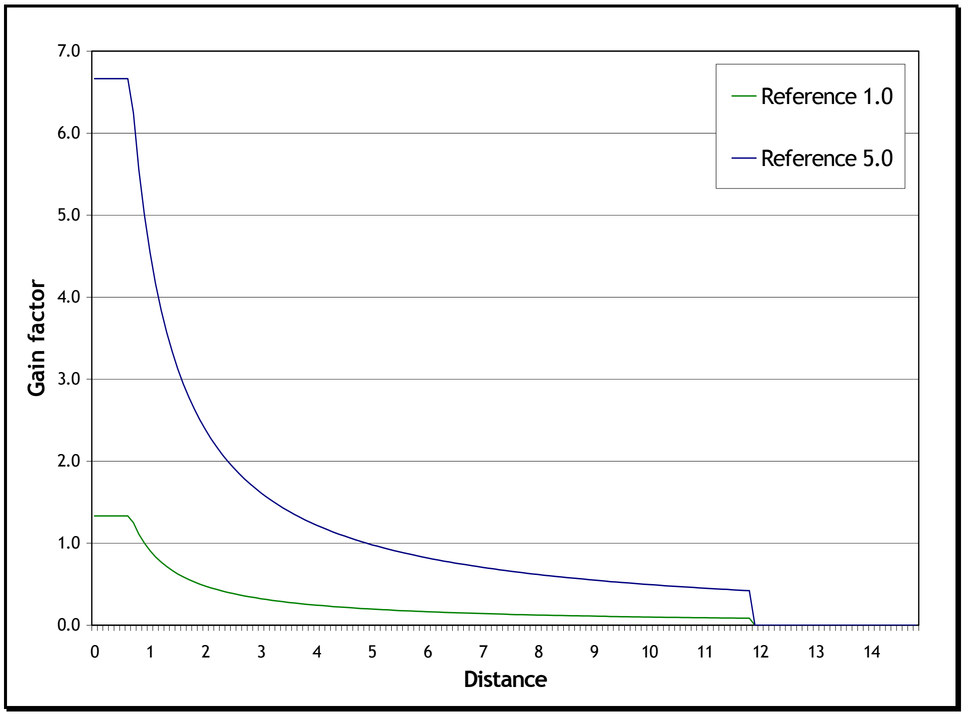

Figure 5: Effect of different reference distances

For an example of choosing appropriate reference distances, consider a jet aircraft and a bee and assume the common practice of using normalized samples. By setting the reference distance of the bee's source to 0.01 units, and the reference distance of the aircraft to 10.0 units, the aircraft becomes 1000 times louder than the bee at the same distance.

6 Source directivity

Many sound sources are omnidirectional, that is they radiate sound equally in all directions so that they sound exactly the same no matter what their orientation. An example would be an exploding bomb. Other sound sources are more accurately represented by a sound source that radiates more sound in one direction than in others, for example a human voice, which projects more forwards than in other directions.

Real life radiation patterns for directional sources are complex, but a good effect can be created using the concept of the "sound cone", the axis of which defines the direction of strongest radiation, as shown in Figure 6. Within a certain angle of the axis (the "Cone Inner Angle"), the sound is not attenuated. As the listener moves outside this zone, the sound is gradually attenuated until an angle is reached at which no further attenuation occurs (the "Cone Outer Angle"). This maximum attenuation is called the "Cone Outer Gain". The developer also has to specify the Cone Orientation Vector that represents the axis.

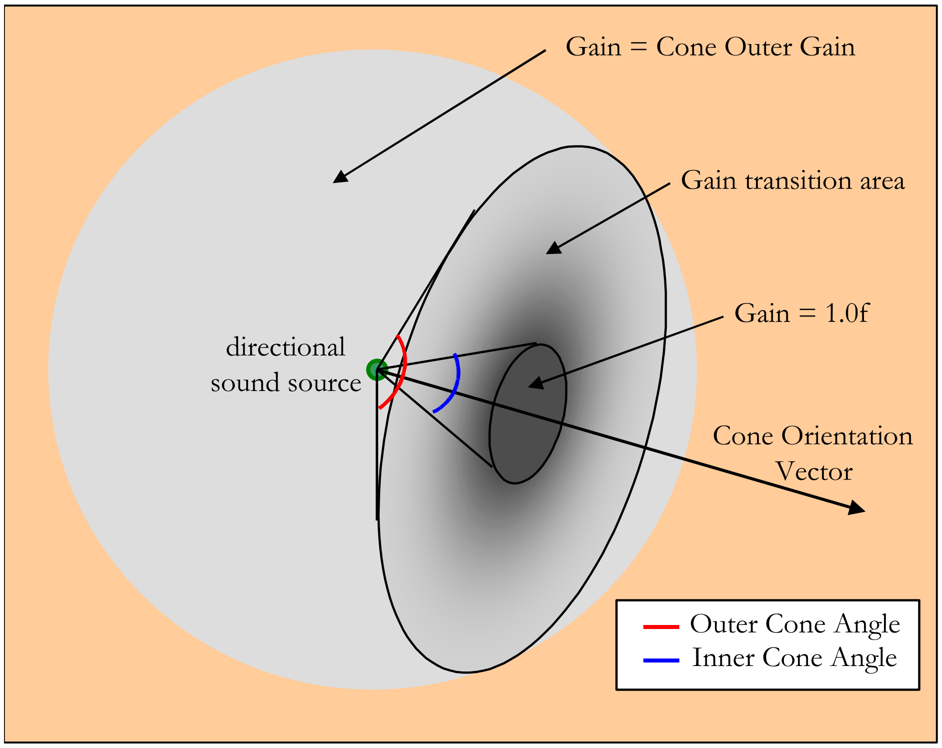

Figure 6: Specifying directivity - the sound cone

If the Cone Outer Angle is not much larger than the Cone Inner Angle, there will be a relatively sudden volume change as the listener moves through these angles. This would be appropriate for some kind of weapon that emits a narrow beam, for example. A useful optical analogy is a lighthouse, which emits a beam of light. The wider the beam, the larger the Cone Inner Angle should be. The sharper the focus of the edge of the beam, the closer the Cone Outer Angle should be to the Cone Inner Angle.

One way to think about the meaning of Cone Outer Gain is to think about the directional and omnidirectional components of the sound source. If a Cone Outer Gain close to 1.0 (or level of 0 dB) is specified, most of the source's power will be omnidirectional. Conversely, if a value close to 0.0 (-100 dB, or silence) is specified, most of the power will be directional.

In GameCODA, sound sources are by default omnidirectional and in the first instance a developer new to GameCODA can ignore the complexities of source directionality and still achieve great 3D audio.

7 Doppler shift

When a sound source is moving at speed relative to a listener its pitch changes. This change is called Doppler shift and is most noticeable if a sound source moves quickly close to the listener. In order to calculate the amount of pitch shift in a real world situation, it is only necessary to know relative velocities of the source and listener, in addition to knowledge of the physical medium through which the sound waves travel, which is usually air. These parameters can in turn be calculated from knowledge of the spatial positions of the sound source and listener as they vary with time.

However, GameCODA does not calculate Doppler pitch shifts from source and listener position because it is usually more convenient for the developer to directly specify the velocities of the source and listener. This allows the developer more flexibility, as these velocities can be decoupled from the actual physical velocities of the source and listener. If desired, the developer can easily calculate the source and listener velocities that correspond directly to their physical movement, thus linking the two again.

It is good practice to use Doppler only on selected sound sources on which the effect is going to be most effective or noticeable, since Doppler processing can use additional memory and processing power. For example, Doppler works well for speeding bullets but not for walking characters.

In addition to source and listener velocities, GameCODA has a global setting that affects Doppler shift for all sources. The pitch shift effect may be increased, reduced or disabled altogether by setting the Doppler factor.

8 MacroFX

The basic concepts of 3D positional audio assume that the effect of altering the distance between a sound source and the listener is only to change its apparent volume. This approximation of the real life situation works well for most scenarios and provides a simple mechanism for distance representation. The assumption is that the listener's head is small compared to the distance between the sound source and listener, and hence that the attenuation due to distance from the source to each of the listener's ears is approximately the same. The simple model therefore applies the same distance-related gain to each ear, by applying the gain to the source itself.

However, when a sound source is very close to the listener, the assumptions built into this approximation are no longer true and this simple model is deficient, as can be seen in Figure 7.

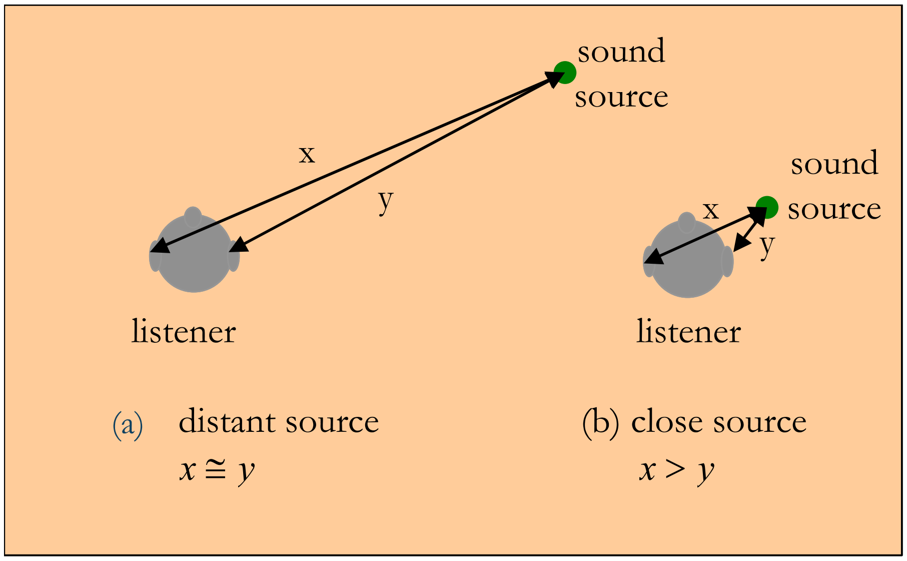

Figure 7: Near field sound sources

Sensaura therefore developed the MacroFX algorithm to model behaviour close to the listener's head more accurately, applying a different gain function to each ear. The white paper "MacroFX Algorithms" describes this aspect of the technology in more detail.

MacroFX does not require any developer effort, since the effect is automatically applied when the sound source is close to the listener. MacroFX works best for quiet sound sources when they move close to the listener. An example is the sound of a buzzing insect, or even a swarm of them.

In order to work properly, MacroFX needs to be able to increase the gain of the signal, so there must be some headroom available to allow this. The developer should therefore set the level parameter of the sound source to less than 0 dB (i.e. linear gain less than 1.0). When set to match normal human hearing, MacroFX requires a maximum headroom of 18 dB. Even greater headroom is needed if the global or per-source MacroFX controls available in the API are used to exaggerate the effect. If a full 18 dB or more of headroom is too much (because it makes a sound source too quiet at distances further from the listener) then not to worry – the MacroFX effect will still be present, but limited by the available headroom.

Setting source levels to values below 0 dB is normal for most game audio where samples are normalized (i.e. recorded at maximum volume and use all of the available bits – usually 8 or 16 – in a PCM sample), with in-game levels balanced through source level and 3D distance model settings. So, this normal practice tends to provide the necessary headroom for MacroFX as a convenient by-product.

9 EnvironmentFX

A sound generated within a particular acoustic environment typically reaches a listener via many different paths. The listener first hears the direct sound from the sound source itself. Somewhat later, he or she hears a number of discrete echoes caused by sound bouncing once off nearby walls, the ceiling or the floor. These sounds are known as *early reflections*. Later still, as sound waves arrive after undergoing more and more reflections, the individual reflections become indistinguishable from one another and the listener hears continuous *reverberation* that decays over time. This combination of direct, reflected and reverberant sound is illustrated in Figure 8.

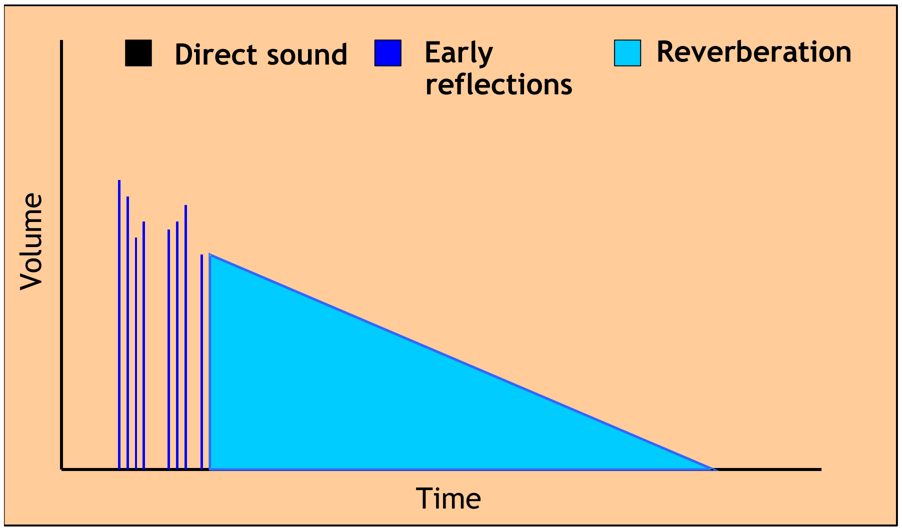

Figure 8: Typical distribution of direct and reverberant sound

Sensaura EnvironmentFX™