3D Audio

Welcome!

This program models realistic spatial audio by animating sound sources in a virtual room. Behold - the power of your desktop converges anew with art and high technology!

For an introduction to spatial perception, we offer the Quick Start.

For details on the program's usage, please do the Tutorial.

What's new with Version 1.2?

System Requirements

Program Status

Contact the Authors

You may contact Climax Software Solutions at :

E-Mail	:	climax@audiophile.com
WWW	:	http://www.audiophile.com/climax
Mail	:	You can get the valid address by writing an email!

Registration Form

Name : _____
Company : _____
Address : _____
_____
_____
_____
E-mail : _____

Your preferred username
for registering 3D Audio : _____

Payment method

[ ] Cash
[ ] Eurocheque (in German marks).
[ ] Direct deposit
[ ] Credit card : [ ] VISA [ ] Diners [ ] AMEX
Credit card no. : _____
Expiration date : _____
Name on card : _____
Billing address : _____

I understand the License Terms completely.

Date : _____

Signature : _____

License Terms

The received password may be used by one single user only. This particular license is confined to private use only; for commercial or institutional use, contact the authors!

Disclaimer of Warranty

THIS SOFTWARE AND THE ACCOMPANYING FILES ARE SOLD "AS IS" AND WITHOUT WARRANTIES AS TO PERFORMANCE OF MERCHANTABILITY OR ANY OTHER WARRANTIES WHETHER EXPRESSED OR IMPLIED. NO WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE IS OFFERED. IN NO EVENT WILL THE AUTHORS BE LIABLE FOR SPECIAL, DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES, LOSSES, COSTS, CHARGES, CLAIMS, DEMANDS OR CLAIM FOR LOST PROFITS, FEES OR EXPENSES OF ANY NATURE OR KIND.

The user must assume the entire risk of using this program.

Choose the fully-qualified path for definition files. It is provided for compatibility and need not be set.

Press to select the definition file path, by means of a file dialog.

Choose the name of the PCM-File to write the computed data to.

Select the output file by means of a file dialog.

Select the monaural, 16-Bit big-endian PCM-File to use as sound data for the current source.

Select the PCM-File to use for this source by means of a file dialog.

Specify the .HTF File containing the head-related transfer function [as impulse responses] to use.

Select the .HTF File to use, by means of a file dialog.

Use the internal set of head responses, instead of specifying an external file.

Plug-in Software Development Kit (SDK)

3D Audio provides an interface to record motion paths from external devices and programs. You can find a demonstration of the possibilities by using the joystick device record mode.

Specify the samplerate of both the impulse data and the source sound files.

Give the length of the internal synthesis buffer. Must be larger than the maximum delay occurring in the scene.

The length of one atomic computation block in samples. Positions and Doppler-shifts are updated every block.

Specify the time that this source is to start, relative to the beginning of the simulation.

Determine the power of the sound assumed for this source. Allows you to adjust the relative volume of the sources.

The path-loss exponent for this source. For free space, this would be -2.0, but smaller values may yield more realistic results.

The order of reflections to compute for this source. Zero means only the direct beam is computed, one means reflections over one wall are computed, and so forth.

This value allows you to change the radiation characteristic of the source, which is given as function of the angle between the direction of emission and the direction of the orientation of the source:

Exp [- EmissionFactor * Angle]

A value of zero yields a constant radiation pattern; values above concentrate the radiation to the direction of orientation.

Define the length that this source is to run in seconds. If the value exceeds the length of the input file, looping is automatic.

Check this to cause the length the source runs to be exactly that of the input file.

Define the size of the box-like room that the scenery is to take place in. Values are given in meters.

The size in X-direction of the box-like room in which the scenery is to take place, given in meters. This value corresponds to the width of the room.

The size in Y-direction of the box-like room in which the scenery is to take place, in given in meters. This value corresponds to the depth of the room.

The size in Z-direction of the box-like room in which the scenery is to take place, given in meters. This value corresponds to the height of the room.

Define the reflection properties of the virtual room. The given values specify the reflection coefficients: a value of e.g. 0.7 means that reflected waves are attenuated by 30%.

Check this to apply a modified computation, which attempts to incorporate the impact of the distance of the source on the used head related transfer function. This is quite the experimental parameter, and doesn't seem to yield a big improvement. But then again, the details are what a 3D Audio experience is all about!

Motion Dialog

This dialog allows you to define atomic steps of continous motion. By creating of sequence of such motions, virtually any complex motion can be implemented.

This node contains a list of the simulation's basic motion instructions which define the path the source (and/or head) will follow.

The angular velocity for turning motion, given in revolutions/sec.

The time that the desired motion is to last.

When checked the desired motion is performed in one single step. Use this to initially position the sources, or to create experimental effects.

When checked, the orientation of the source will always be directed towards the virtual head, regardless of the individual paths of motion.

Define the end-point of the motion, by providing its destination coordinates in the X-, Y- and Z-planes.

Describe an absolute translation, i.e. a movement to a given point. The specified point will be proportionally reached during the defined motion time.

Define the speed of the motion as velocities in X-, Y- and Z-coordinates.

Define the the acceleration of the motion in X-, Y- and Z-coordinates.

Define a relative translation, i.e. a motion with a given speed and/or acceleration.

Define the angle that the orientation of the source is to be turned to. The angles are given as "theta" [the angle between the direction vector and the XY-plane] and "phi" [the angle in the XY-plane between the X-axis and the direction vector, measured counter-clockwise].

The angular speed of the rotation of the direction vector. The angles are given as "theta" [the angle between direction vector and the XY-plane] and "phi" [the angle in the XY-plane between the X-axis and the direction vector, measured counter-clockwise].

The angular acceleration of the rotation of the direction vector. The angles are given as "theta" [the angle between direction vector and the XY-plane] and "phi" [the angle in the XY-plane between the X-axis and the direction vector, measured counter-clockwise).

Define a relative rotation, i.e. a rotation with given speed and/or acceleration.

The ciruclar motion of turning will take place on a plane in the three-dimensional space. The point defined by this (X,Y,Z) vector specifies the center of that circle when projected to the plane of motion.

The ciruclar motion of turning will take place on a plane in the three-dimensional space. This [X,Y,Z] vector in effect defines the plane, because it will be located perpendicular to that plane.

Turning means moving the source with a given rate on a circular path, perpendicular to a given axis. The latter is defined by direction and position vectors.

Absolute Rotation means turning the source's direction to a given absolute angle.

Define an absolute translation and rotation, i.e. a movement to a given point and a given absolute angle.

How Do I Register?

The price for registering 3D Audio is an incredible bargain: US-$ 30 for a single-user license. You will then receive a password which will unlock your copy of the program. This license is limited to **private use only!** For commercial or institutional use, contact the authors.

There are two ways to register the program:

Submit the completed registration form to us. Payment may be done by credit card, and users residing in Europe may alternatively send an eurocheque for the equivalent of US$30 in any currency. Payment may also be directly transferred to a bank account - please contact us for the account number. Money transfer happens to be the simplest and cheapest way for users in Germany.

Alternately, you may also use the KAGI registration service for easy and safe registration via the Internet, fax or mail. Kagi resides in the USA, which makes it the preferred choice for American users.

Enter your name as given when registering your copy.

Enter the password you received for your username.

The button for greedy people who will surely rot in Voltaire's best-of-all-possible-worlds...

Press this button to register your copy of 3D Audio, after entering your name and the received password.

Press this button to view the terms for registering.

Pess this button to start the simple and safe Kagi Registration service!

Select the way the Direct3D output is displayed. Possible options are either in a window or in full-screen mode.

Check this to have the Direct3D output displayed in a window.

Check this to let the Direct3D output be display full-screen. The combo box to the right may be used to select the desired resolution and color depth.

Select the desired resolution and color depth for full-screen display.

Define the render mode used by Direct3D.

Renderer output will be a wireframe model.

Renderer output is unlit, flat mode.

Renderer output will be in flat mode.

Renderer output will be Gouraud mode; this looks best but is the most demanding in terms of computational complexity.

The mode of color emulation used by Direct3D.

Emulation mode will be "ramp". This is faster than RGB and still looks reasonably good.

Emulation mode will be RGB. Very heavy load for the computer, but the visual results are optimal...

This window gives you an idea of how the selected Direct3D options will look like when the program is actually running. It can also be used to test your textures, if you use them to help keep your associations of floor, walls, and ceiling consistent. Failure to do this can result in nausea! ;)

If this display is active (click with the mouse on it), you can influence the movement of the displayed room by means of the num-locked keypad and <Pos1>, <End> and <Insert>.

Here you may change the textures used for walls, floor and ceiling of the virtual room. It is easier for one to keep track of a developing scene by using textures that you will naturally associate as being the ceiling and floor, as well as the cardinal directions if possible.

The selection of textures is not global, however, and is the settings are saved with the project file. Be sure to maintain the integrity of your source image paths, or else when you open the project later the previews will look different.

Select the wall that you want to change the texture of.

Brings up a file dialog where you may specify a .BMP-file to use as new texture for the selected wall. Note that the dimension of the picture must be a power of 2, and that tiling is automatic!

Check this to revert back to the default texture of the selected wall.

Select the object that you want to modify and define a new model for it.

Use the combo box to select the object that you want to modify by its name.

Brings up a file dialog which lets you specify a new Direct3D model (.X-file) to use for the selected object.

Check this to use the default model for the selected object.

Lets you change the color of any of the used Direct3D objects.

Displays the currently selected color of the current object.

Brings up a dialog box for selecting a color for the current object.

Check this to revert back to the default color for the currently-selected object.

Use this notebook to select that property of the object which you want to edit. You may modify the spatial position, orientation and scaling of the used Direct3D model to make it fit for later display purposes.

Specify a [X,Y,Z] vector to use as an offset for the currently selected object.

Specify a [X,Y,Z] vector with scaling factors that will be applied to the model.

Specify a [X,Y,Z] vector of angles to which the Direct3D object will be turned prior to display.

Check this to revert back to the default transform settings.

Press this button to access a window for entering your registration code.

Press this when you're bored enough...

Do the tutorial to learn how to use 3D-Audio!

Information about how you can register the program

Here you can easily and safely register this program!

Define a name to identify the new source.

Project Window

This window displays the parameters and settings of the simulation using a tree-like structure. One may dive into the structure and edit the nodes' parameters with their associated context-menus (right mouse-button), or by using the keyboard. Use 'Return' for the node properties, 'Insert' for inserting new sources/motions and 'Delete' to delete sources or motions!

The hierarchical structure groups project data into the following entries:

Global	Contains basic data, such as the name of the output file
Room	Contains the parameters that define the virtual room
Head	Bundles data defining the position and animation of the virtual head
Source	Contains all data defining a sound source. There can be a list of such source-entries, one for each source appearing in the simulation. To insert and delete sources, use the context menu (right mouse-button) or keyboard when a source entry is selected.

This node contains basic data concerning the simulation, such as the output file name, the sample rate of the data, etc.

This node defines the virtual room in which the scenery will take place.

This node contains information on the virtual head, such as its position and motion in time.

Contains basic information on the virtual head; actually it is just a flag which defines if the head is moved during the simulation or not.

Signals if the virtual head is animated and thus must be updated during the simulation. This flag cannot be changed by the user, but automatically changes with the definition of the "motion" node.

This node contains the data defining a sound source.

Contains basic information on the source, such as the PCM-file associated with it, the power etc.

Contains the definition of the motion of the source or the head during the simulation.

Signals if the source is animated and thus must be updated during the simulation. This flag cannot be changed by the user, but automatically changes with the definition of the "motion" node.

A dummy entry which signals the end of the list of sources.

Each node represents one motion instruction where the order of the statements gives the order of the complete, compound motion that will finally be performed. There are various types of motion supported.

The parameters of the parent motion instruction. The type of possible parameters depends on the type of motion selected.
Left-click to select, and then right-click for the list of available options!

A dummy entry which signals the end of the list of motion instructions.

Choose a plug-n from this list!

Here you can get some information about the plug-in.

Check this option for polling the external plug-in at the chosen rate.

Check this to do a step-by-step record.

If you check this the entire control of the recorded motion is up to the plug-in.

Insert the poll interval, in milliseconds. This indicates how often the program checks the input sources for new information.

Insert the default step to step time interval.

With this button you start/stop the recording.

Press this button to record the next step.

Press this button to delete the complete recorded motion list.

Displays the abolute recorded time.

Displays the number of recorded steps.

Here you can see the recorded motion events. You can choose an event and edit it.

Press this button to delete the choosen motion event.

While inserting much data this display shows the progress of the action.

This field shows you the official Name of the plug-in.

This field contains the version of the plug-in.

This field contains copyright information from the company that provides the plug-in.

If this box is checkable, the plugin provides real-time recording.

If this box is checkable, the plug-in provides step recording.

If this box is checkable, the plug-in can take full control over the record behavior.

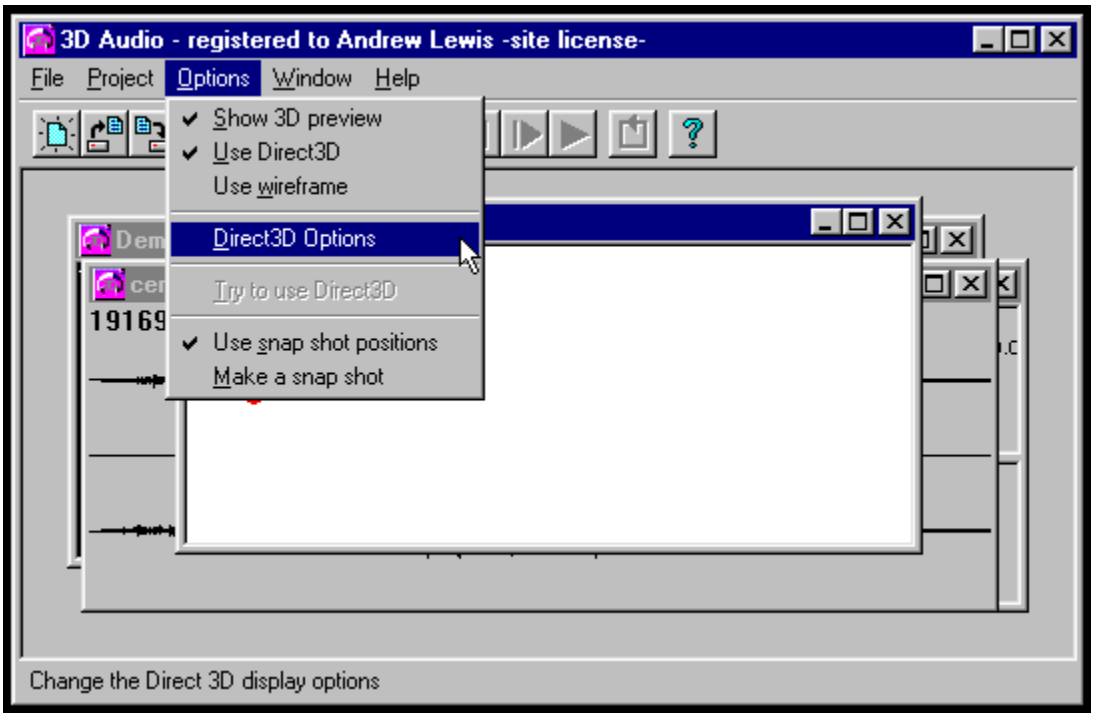
Here you get miscellaneous information about the plug-in.

Project Properties Dialog

This dialog lets you control the basic properties of the various objects such as the room, the virtual head and the sound sources. The actual elements of the dialog depend on the type of object selected.

Options Menu

This menu contains several entries which let you control how the rendered scenery will be displayed on the screen. These submenu entries are available:



Show 3D Preview Disable this to hide the 3D preview output while rendering. This will speedup the render process!

Use Direct3D Check this to let the output be displayed using Direct3D. Naturally, Direct3D must be installed on your system.

Use Wireframe Check this to let the output be a simple wireframe model.

Direct3D Options Brings up a dialog to define a number of parameters when using the Direct3D display, such as desired emulation and rendering modes, or the models used for representing sources and the head.

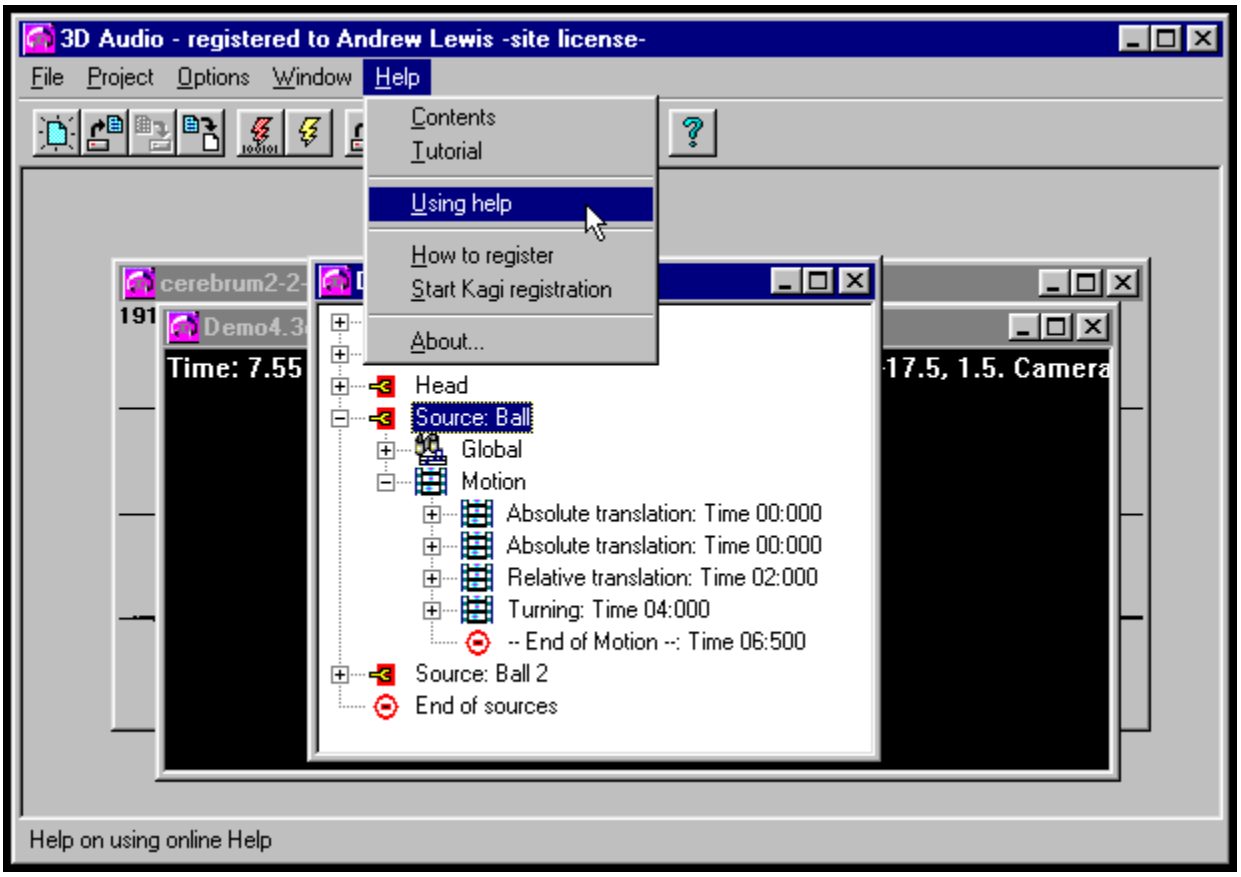
Try to Use Direct3D Instruct the program to check for Direct3D. Invoke this operation if you installed Direct3D after 3DAudio.

Use Snapshot Positions Use the positions of the MDI-windows stored with "**Make a snapshot**" as default when opening new windows.

Make a Snapshot Store the current position of the MDI-child windows.

Help Menu

This is the place for more information on how to use **3D Audio**.



Contents Take you to the top of the helpfile!

Tutorial A tutorial how to use 3D-Audio!

Using Help How to use windows-style help files

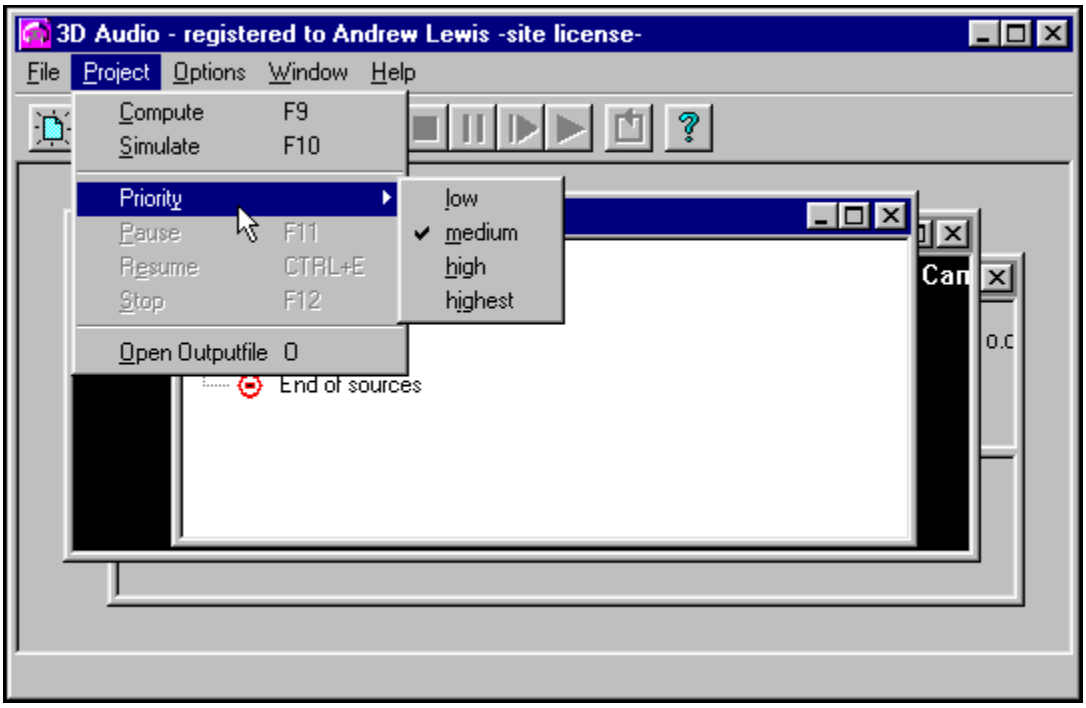
How to Register Brings up the instructions on how to Register 3D-Audio!

Start Kagi Registration Register the program easily and safely with the Kagi registration service located in the USA!

About About 3D-Audio! Here you can insert the registration information and obtain your personal copy!

Project Menu

The project menu contains the following items which invoke a simulation and control its execution (the equivalent key being given in parentheses):



Compute (F9) Start computing the current project. If another computation is in progress, it will be canceled. You may open another project and work on it, or begin working within another application. Sometimes it helps to open the applications you'll be needing before one begins a long 3D-Audio rendering session.

Simulate (F10) Start simulating the current project. This means that the scenery will be rendered and displayed, but no audio data is to be computed. This can be used to verify the motion sequences.

Priority Enables you to adjust the priority of the thread computing the audio data, relative to the other processes already running.

Pause (F11) Pause the current computation.

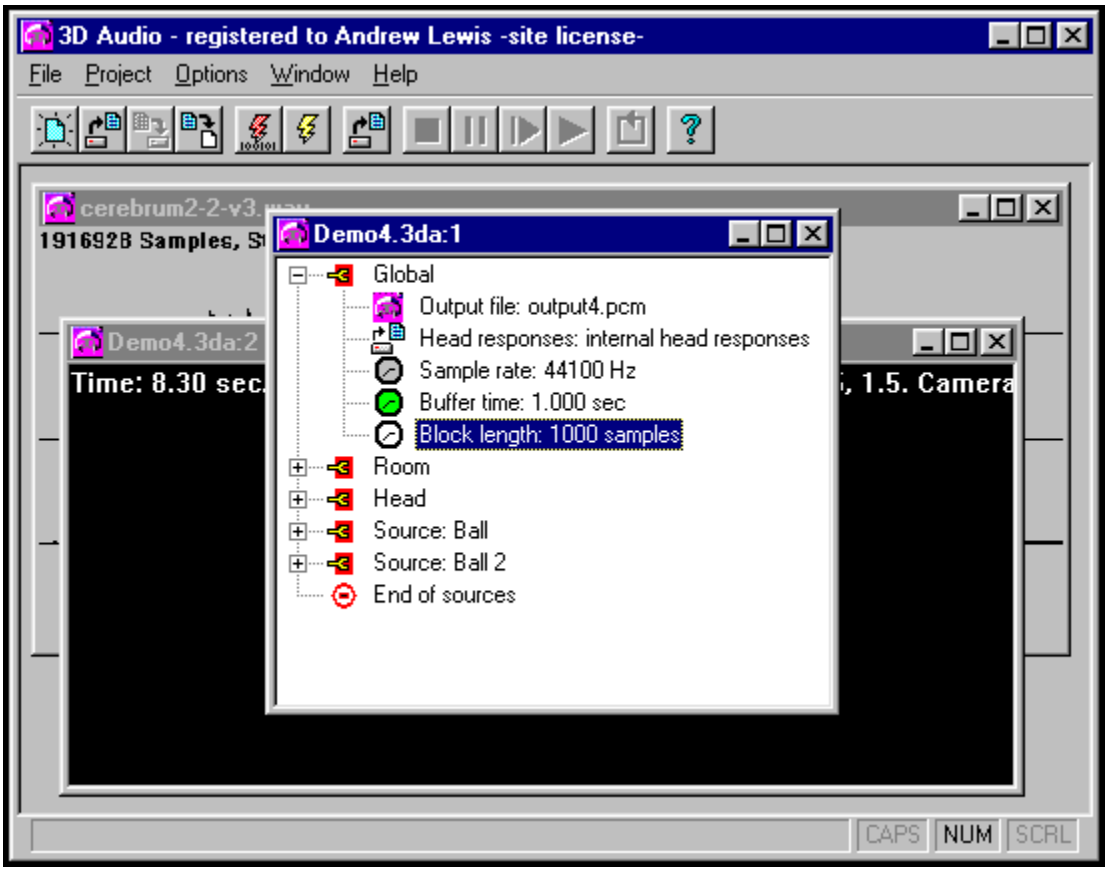
Resume (CRTL+L) Resume the paused computation.

Stop (F12) Stop the current computation.

Open Output File (O) Open the PCM-File that contains the output data; this may even be performed while the simulation is running.

Main Menu

The contents of the main menu depend on the type of MDI-child that is currently active. These submenus are defined:



File — The file menu contains items that allow you to load and store simulation project files, sound files, and your Wizard preferences.

Project — (only available for project and render display window)

Audio — (only available for audio-waveform display window)

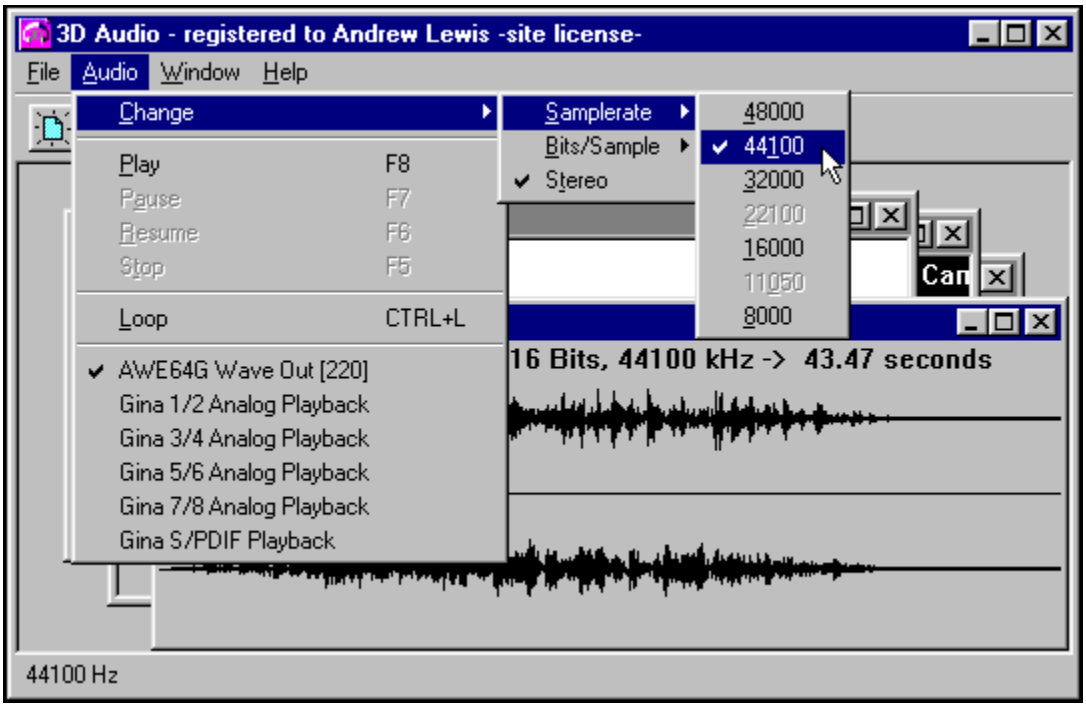
Options

Window — Arrange or close all your windows. Very standard!

Help

Audio Menu

The audio menu is available when a waveform window is the current MDI-child. It contains these entries:



Change

Play (F8)

Pause (F7)

Resume (F6)

Stop (F5)

Loop (STRG+L)

The entries of this menu allow you to change the parameters of the current waveform, such as samplerate, bits per sample and mono/stereo. Note that changing doesn't mean the data are changed, but only the assumed format! These parameters may be changed:

Sample Rate Available are the usual rates, from 48KHz to 8KHz.

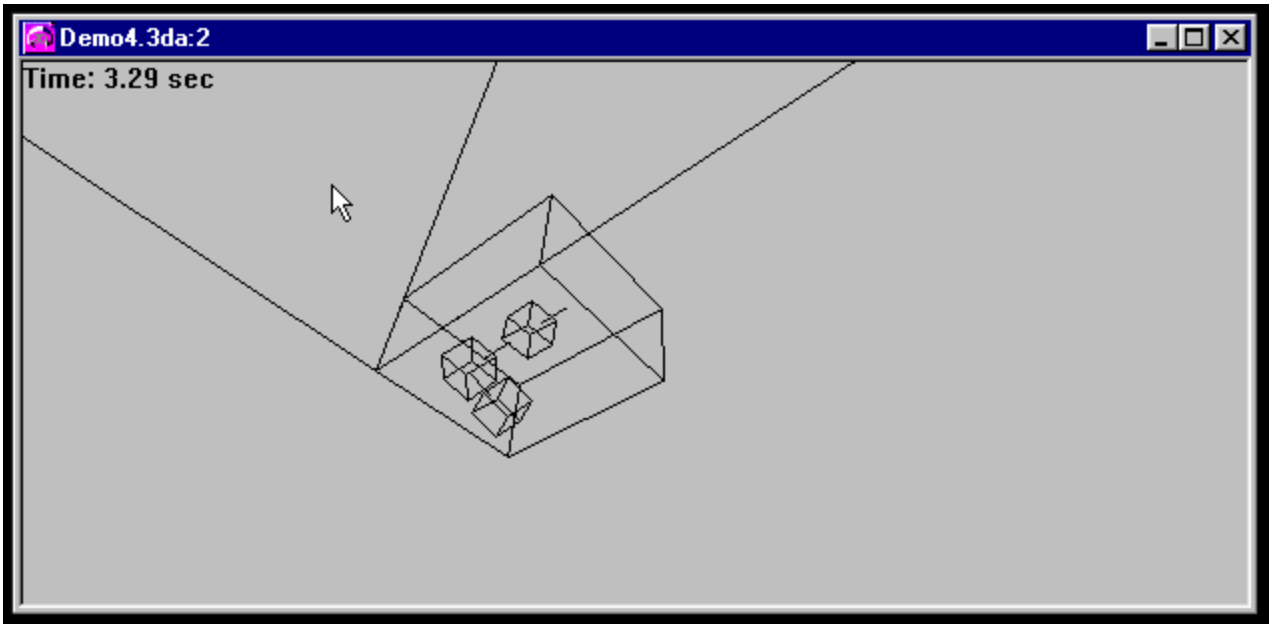
Bits per Sample May be either 8 or 16; Keep it at the higher resolution for best results!

Stereo Check this when using stereo waveforms.

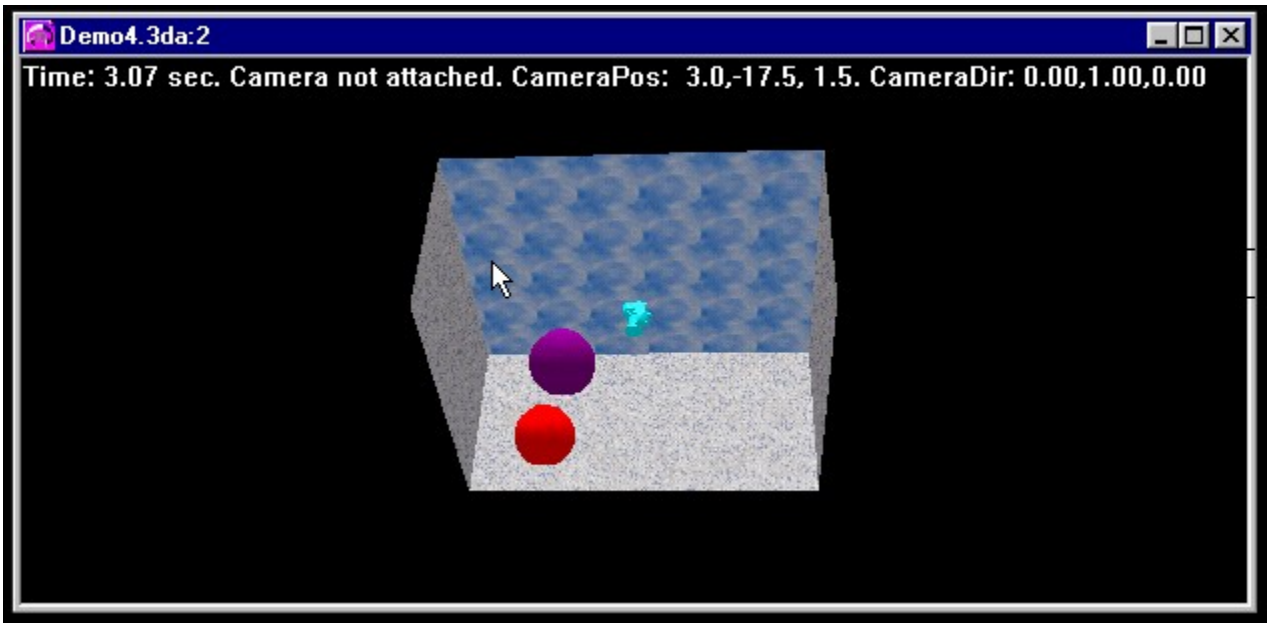
Visualization Window

This window displays the scenery you have created. The rendering will either be done by Direct3D, or it will be a simple wireframe model. In either case, you may switch the viewpoint interactively.

Wireframe View



Direct3D View



Press the left mouse button within the window, and drag the mouse to change the view. In addition, the following keyboard shortcuts are supported:

<PgUp>	Rotate the viewpoint around the center of the room by a small amount.
<PgDn>	Rotate the viewpoint around the center of the room by a small amount, in the opposite direction.
<Pos1>	Start rotating the viewpoint around the origin of the coordinate-system.
<End>	Start rotating the viewpoint around the origin of the coordinate-system, in the opposite direction.
<Delete>	"Freeze" viewpoint, i.e. stop moving, rotating etc.
<Insert>	Reset the viewpoint to the default.
<Csr-Keys>	Move the viewpoint in space.
<0>	Attach the viewpoint to the head.
<1..9>	Attach the viewpoint to source 1..9 if they exist.

PLEASE NOTE that when navigating within the preview window of the "Direct3D Options" dialog, the cursor-keys are not available. Use the num-locked keypad instead.

Audio Window

This window displays the waveform of a loaded audio file. When this type of window is the active MDI-child, the audio menu is available, which allows to play the sound etc.

You can save the audio data as raw pcm-data or as a windows pcm WAV-file!

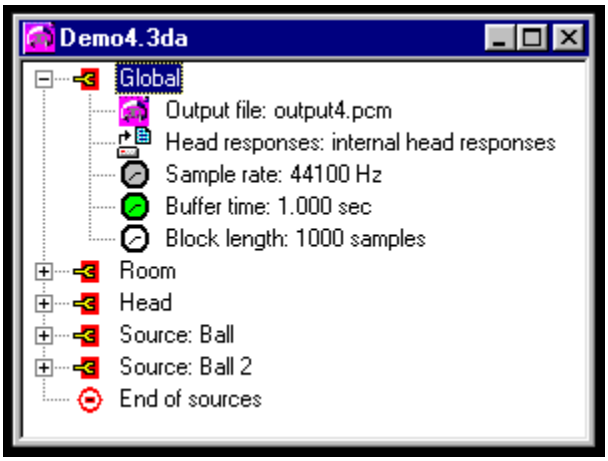
Progress and Verbose Window

This window displays the current spatial position of the objects during computation. Additionally, progress bars indicate the current degree of completion for each of the sources.

During initialization of the computation, the upper part of the window serves as log-display to dump error messages if they occur.

Entry "Global" in the Project Window

This entry contains the generic parameters of the project. The following items are found here:



Output File The name of the output file in PCM format.

Head Responses The set of measured HRTF data to use. Should currently be the internally-stored set.

Sample Rate The sampling rate of the audio data. Must be 44.1 KHz for the internal set of HRTF data.

Buffer Time The length of the interal audio buffer for calculations. This length must exceed the maximum delay that occurs in the simulated scene.

For example, if one creates a room of 10x10x5 meters and uses reflections of the second order, the maximum path length from source to head is:

(max. reflections + 1) * max. path length without reflections.

This also happens to be the longest diagonal line in the room. The example works out to about 40-50 meters, which means a maximum delay of about one sixth of a second.

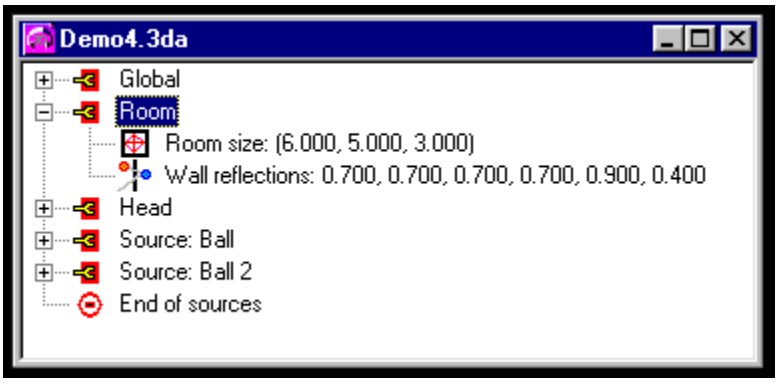
Block Length The interval, in samples, during which the positions and doppler shifts of the source sounds will be updated.

The default is 1000 samples each, or about once every 20 milliseconds. This seems to be a good value; you only may need to lower it when using very fast animated sources.

WARNING: do not choose values below 128 samples!

Entry "Room" in the Project Window

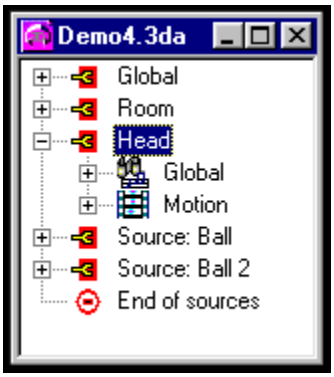
This entry contains the parameters of the virtual room.



Room Size	The size of the room in meters.
Wall Reflections	The reflection coefficients for the walls of the room.

Entry "Head" in the Project Window

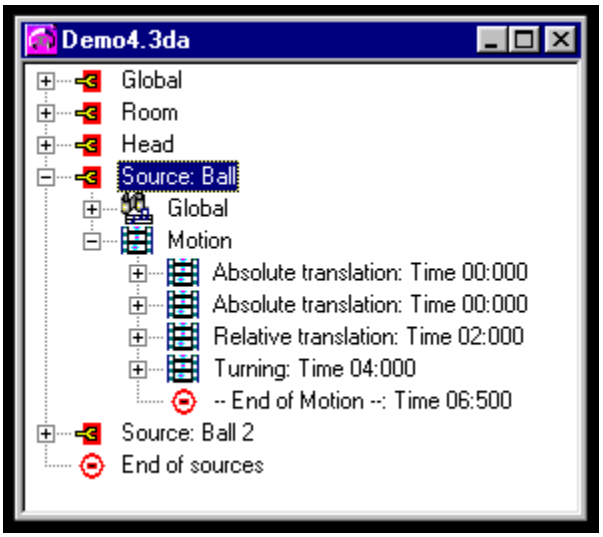
This entry contains the its animation instructions for the virtual head, which define its position and orientation during simulation.



See the description of the Source / Motion Entry for a more detailed explanation of these parameters.

Entry "Source" in the Project Window

This entry contains information defining a sound source and its simulated motion. There are two sub-entries concerning these definitions:

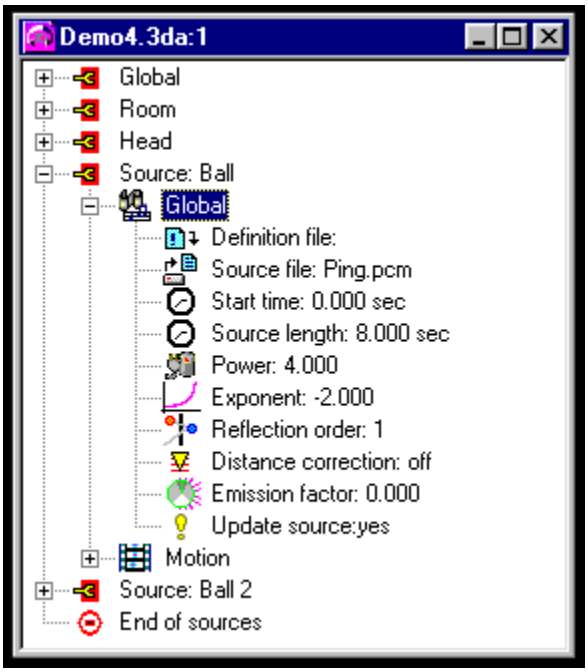


Basic Contains the generic definition like the name of the sound file associated with the source.

Motion Contains the motion instructions.

Entry "Source / Global" in the Project Window

This entry contains the generic definition data about the source:



Definition File A relic from an older command-line version. Simply ignore it!

Source File The name of the sound file to use as output of the source.

Start Time The time to elapse before starting the source.

Source Length The length of the sound that the source emits in seconds. If the length exceeds the length of the according PCM-File, the sound will be looped.

Power Factor to amplify the sound of the source with. Can be used to adjust the volume of the sound arriving at the head. Note that with higher reflection orders (see below) the volume of the arriving sound automatically increases.

Exponent The path-loss exponent used for computing the attenuation of the arriving sound, depending on the distance between source and head. The attenuation will be proportional to the distance to the power of the given exponent. The natural value is -2.0, but it may lead to a too strong attenuation especially when using few reflections. In such a case you might want to lower the value to about -1.5.

Reflection Order The order of reflections to allow for this source.

DistanceCorrection A rather experimental option, which is intended to allow for the distance-dependency of the HRTF-data.

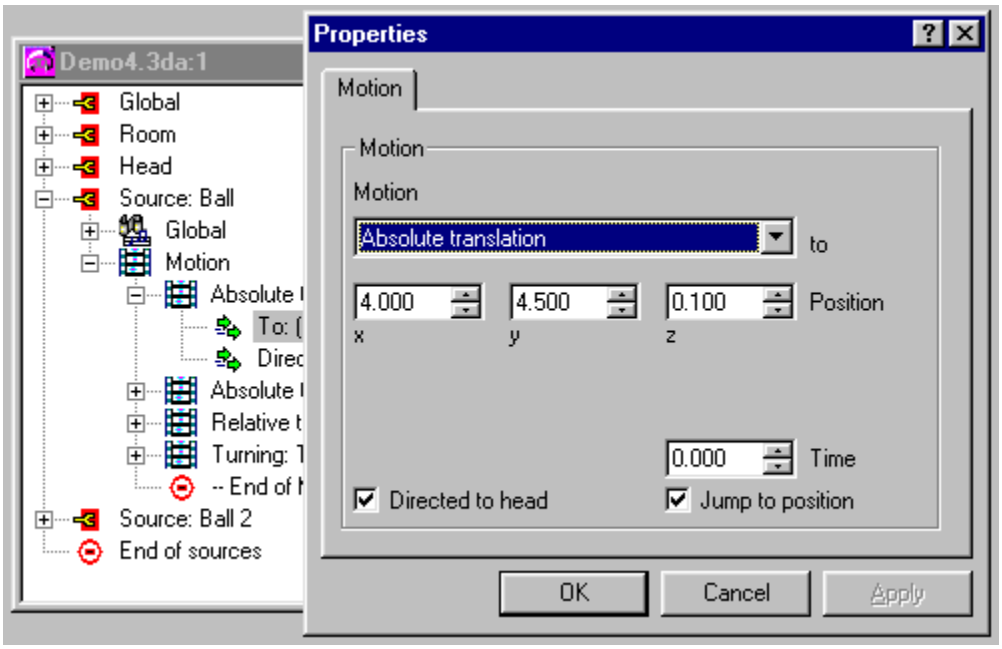
Emission Factor Controls the radiation pattern of the source. A value of zero yields an

omnidirectional emission. Higher values concentrate the radiation more and more to the direction and orientation of the source. Practical values may be in the range of up to 0.7, which already yields a very strong concentration of radiation.

Update Source Cannot be modified; it indicates if the source position will be updated during computation.

Entry "Source / Motion" in the Project Window

This entry contains the definition of the motion during simulation. The sub-entry "**Motion**" of the entry "**Head**" is identically structured.



The entry contains a list of sub-entries which define the path that the source (or head) is to follow during simulation. Like a translation, each of the sub-entries defines one type of motion.

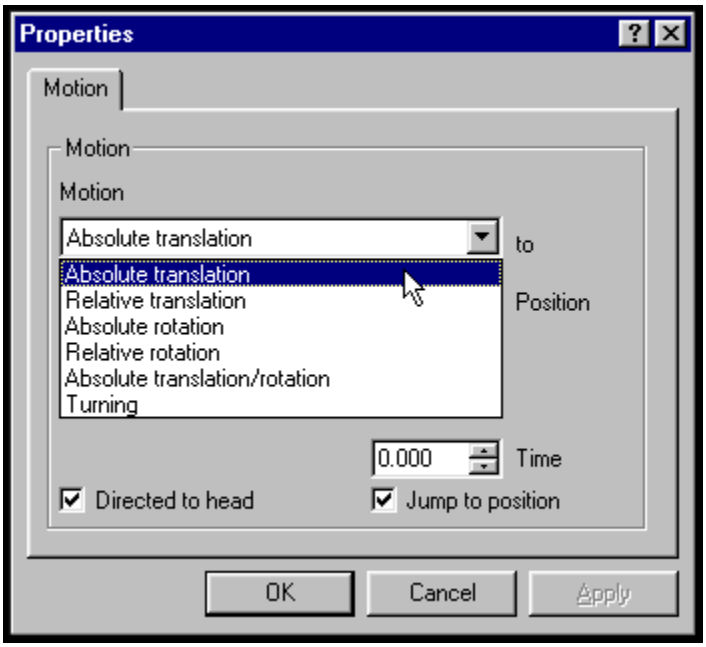
New subentries may be added by selecting "**Insert**" within the context menu of any one of the existing subentries, and the newly created entry will be inserted before the selected one. For the use of the motion list, please see Section Three of the Tutorial.

Each motion instruction in the list is made up of the type of motion to perform, including the parameters for this motion type and the time that the motion is to last.

One may, however, specify that the motion is to be performed immediately, that is in 0.00 seconds' time. This is called "**Jump to Position**" and is mainly useful for the initial placement of sound sources. To enable this feature, check the appropriate box in the associated Properties dialog.

Moreover, in this dialog you may specify that the orientation of the source remain directed to the head's position during movement (called "**Directed to Head**"). Please note that "**Directed to Head**" is only available when moving sources; when the head is moved this option is impossible.

Currently, the following types of motion instructions are supported:



Absolute Translation: Moves the source to an absolute position within the virtual room. Leaves the orientation of the source unchanged, unless "Directed to Head" is checked (see above).

Relative Translation: Moves the source with a given speed and acceleration. Leaves the orientation of the source unchanged, unless "Directed to Head" is checked (see above).

Absolute Rotation: Rotates the orientation of the source to an absolute angular value. Leaves the position of the source unchanged.

Relative Rotation: Rotates the orientation of the source with a given angular speed and acceleration. Leaves the position of the source unchanged.

Turning: Moves the source on a circular path around a given center and axis. Leaves the orientation of the source unchanged, unless "Directed to Head" is checked (see above).

Start playing the current waveform.

Pause playing the current waveform.

Resume playing the current waveform.

Stop playing the current waveform.

Select loop mode when playing a waveform.

PCM Files

PCM-Files contain sampled audio data. The format of such files expected by this program is:

- Sampled at 44100 Hertz
- Monaural
- 16 Bit 'Big Endian' [i.e. the Wintel byte-ordering scheme]

If you wish to use a sound file in another format, check out the variety of third-party products such
as "CoolEdit" or "SOX". These and many others can convert your data into raw *.PCM audio.

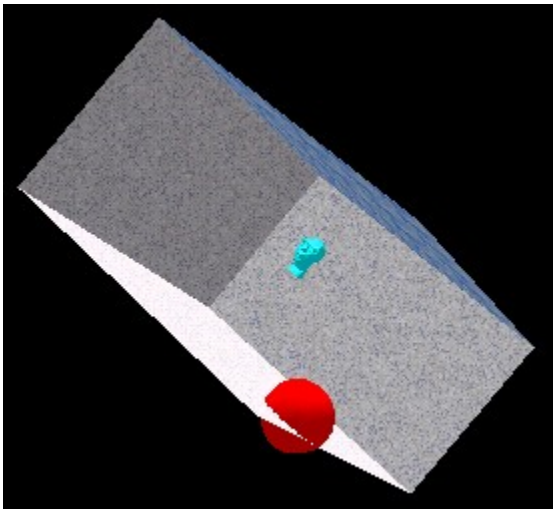
Source Files

As source files, you can use raw .PCM or .WAV-files. The format of such files expected by this program is:

- Sampled at 44100 Hertz
- Monaural
- 16 Bit

If you wish to use a sound file in another format, utilize third-party products such as "CoolEdit" or "Sox" to convert the data. There are many such programs available!

Quick Start



3D Audio enables one to create sound files that convey true spatial audio placement. The software supports multiple sound sources, and features accurate physics that can even surpass reality if desired.

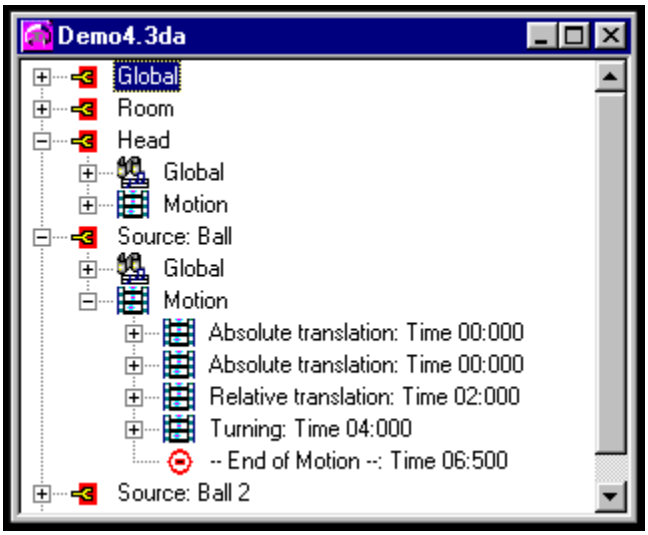
The prerequisite, however, is that for full effect one must listen to the rendered output over headphones. Although the effect is discernable over loudspeakers, it is with headphones that a direct link is made between **3D Audio** and your ears!

The program focuses on the various empirical influences which determine the way our brains localize audio. That is, the most essential of these are modeled during simulation. Since the real world envelops us with objects that emit and reflect sound waves, **3D Audio** must likewise represent the physical surroundings of these sources. This is done by introducing a virtual room.

Within, a virtual head and an arbitrary number of virtual sounds may be positioned. The reflections from virtual walls are calculated according to the order of reflections specified by the user. And since static scenes are a tad boring, one may freely animate both head and source!

If one has Direct3D installed, the animated scene may be previewed as a rendered picture sequence; otherwise the project is previewed as a simple wireframe animation. Either is quite useful for the verification of motion commands, and may be simulated without computing the audio.

To start, load one of the demonstration projects. Examine the simulation parameters by delving into the tree structure.



When satisfied, render the scene with "**Project / Compute**". Access the result with "**Project / Open Output File**", and have a listen.

Then, start changing the parameters around to gain a feel for them. The usage of the tree-structure should be intuitive to Windows 95 users; one may change the parameters by double-left-clicking or with the right mouse button into the associated context-menu.

PLEASE NOTE that some operations such as inserting new sources and motion-capture are only available via this context-menu.

To get more familiar with **3D Audio**, please review the Tutorial.

Crosstalk

When a binaural signal intended for the right ear collides with that of the left, mental confusion is caused due to phase cancellations [a.k.a. comb filtering]. In addition, as the signal bounces around the listening room the carefully-organized spatial information becomes jumbled.

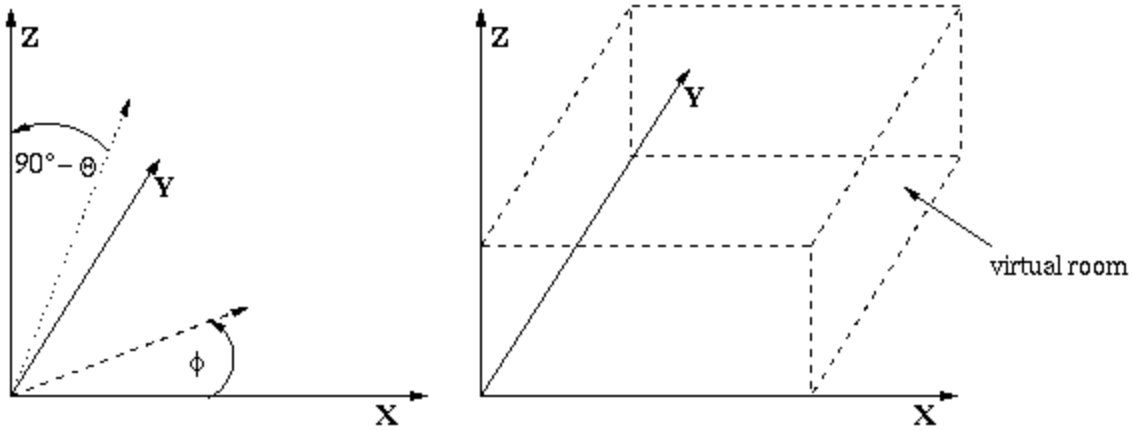
The result is the collapse of the binaural image.

Definition influenced by:

Ambiophonics, by Ralph Glasgal and Keith Yates
Ambiophonics Institute Press, ISBN 0-9646634-0-6

The Virtual Room

The simulation environment is modeled to have a regular, 'boxlike' shape. To enhance realism, the interior dimensions and reflecting properties of each surface may be defined. The figure below illustrates the definition of the angles and the orientation of the room [within the coordinate system]:



The **reflection coefficients** of the walls may be given in the range from 0.0 to 1.0, where zero means total absorption and one means total reflection. Thus, a wall may be omitted by setting its coefficient to zero. To simulate a surrounding with just a floor, set all coefficients except the floor's to zero.

Since it would be nice for the rendering time to be finite, one may specify the **reflection order** to incorporate. A value of N means convoluting the line-of-sight wave along with indirect waves arriving from a maximum of N walls:

Reflection Order	Reflected Beams
0	0
1	6
2	24
3	62
4	128
5	230
6	376
7	574

That is, each higher order of reflection introduces additional waves from various directions; therefore, each wave must be processed using a different impulse response.

Hence, rendering time dramatically increases with N!

Using reflections of first order significantly increases the simulation's "naturalness". Second-order reflections perform a little better, and quadruple the rendering time. For now, it seems that seven is the highest order of computation that completes in finite time.

The results of seventh-order reflections sound really great - and on a modern PC, rendering takes 2000 times the project's duration to complete...

Localization of Sounds

There are a number of influences that support our spatial perception of sound. Although the actual processes are not fully understood, the academics have thus far identified the following four cues as being the "**Head-Related Transfer Function**":

Interaural Delay: The time difference between a sound reaching the closer and farther ear. This delay is zero for sources directly behind, ahead or above - and reaches a maximum of about 0.63 ms for sounds from the far left or right. The exact amount depends primarily on the direction of the sound, but is also influenced by distance and frequency.

Head Shadow: A complex effect; imposed on a sound having to pass through or around the head to reach the farther ear. In addition to the attenuation in amplitude, there are many other factors such as distance and frequency content of the sound, head mass, and reflections of increased prominence from the 'far side' walls. **3D Audio** accounts for all of this, with the quality factor being determined by the order of reflections!

Frequency Response of the Outer Ear: Significant for frequencies above 4kHz, and critical for the intuitive location of unseen sound. This is largely because the brain has innately tuned its spatial associations to the unique frequency response of the outer ears.

Reflections from the Shoulders: Relevant for frequencies below 3kHz. Also influenced by the sound's orientation.

This program empirically models the spatial impact of the above. There are also several other 'spatialization cues':

Vision: We so strongly depend on sight that virtually any aural evidence of a source's position is superceded when faced with conflicting visuals.

Early Echoes and Reverberation: The effects of direct and indirect sound; primarily a characteristic of the surrounding environment.

Head Motion: Subconsciously performed to gather information through subtle re-alignments of the head. Naturally, this cannot be implemented in **3D Audio**!

Parts of the above have been taken from:

D.A. Burgess. "Techniques for Low Cost Spatial Audio". Tech Report, Georgia Institute of Technology.

3D Audio's Model of Spatial Perception

The most important influence upon the spatial perception of acoustic signals is attributed the term head-related transfer functions. These are modeled empirically within the program, as represented by a set of impulse responses measured by microphones placed inside a 'dummy' head.

The measurement technique used is known as binaural recording. While the details are beyond the scope of this help-file, binaural audio ranks high among surround professionals as being a very natural and accurate method of spatial audio reproduction. Since the microphones' placement simulates real ears, the output should likewise be fed **directly** into real ears for the fullest effect.

Otherwise, the signal will become corrupted by crosstalk and acoustic coloration from the listening room. Likewise, the listener's brain will be hit with conflicting pinnae cues which reduce his ability to accurately perceive spatial information. If speakers are all that is available, however, set them up level with the head and aimed directly at the ears.

Since each individual response strongly depends on the relative direction of the sound, the included set of impulses covers the majority of possible directions. Applying these data to a sound file, i.e. convoluting an impulse response corresponding to a certain direction with the sound data, yields a vague impression of spatial positioning of the sound.

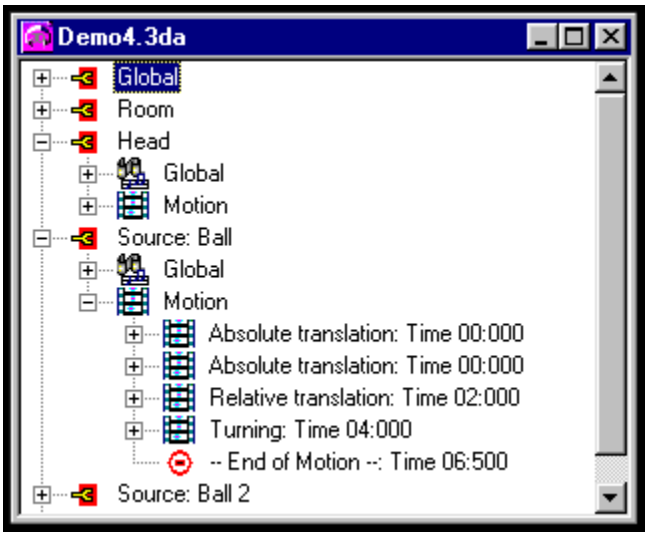
This vagueness is due to the fact that the response measurements took place inside an anechoic chamber, which is perceptually confusing (and odd in general). Therefore, a virtual room has been introduced to allow for signal reflections from the walls.

Providing that the sound intended for the left ear reaches only the left ear and likewise for the right, these reflections supply both additional information for localization and the substance for a more realistic impression. Cool, eh?

Animating the Objects

OVERVIEW

Objects, which are defined here as the "Virtual Head" and associated "Sources", may be given a set of instructions describing the motion to be simulated. These movements concern both the position and the orientation of objects.



Each object is initially positioned in the origin of the Cartesian coordinate system, and oriented along the X-axis. The motions are perfomed as a timed sequence, and if the specified time is zero that particular motion will occur in one atomic step.

This latter usage is primarily for the initial positioning of objects. [One may also, however, create interesting effects - so be sure to experiment!]

Animation instructions may either be inserted 'by hand' or through Motion Capture.

COMMAND REFERENCE

Absolute translation, which means specifying an end-point to be reached at the given time, with the object by moving along a straight line.

Relative translation, which means giving a speed and optionally, an acceleration. The object will move according to these values for the given time.

Absolute rotation, which means dictating an end-orientation for the object. This orientation will be reached after the given time. Note that rotation implies no change of position!

Relative rotation, meaning changing the orientation of the object with a given angular speed and acceleration. Note that rotation implies no change of position!

Turning, which means animating the object on a circular path, located in plane which is defined by an axis perpendicular to it. The center of the circle is given by a vector projected onto the above-mentioned plane.

USAGE NOTES

When applied to a source all possibilities, except instructions on rotation, may be set such that the orientation of a source will always point towards the virtual head during the motion. This is accomplished with a simple check-box.

Normally, a translation leaves the orientation unchanged. Changing of the orientation naturally only affects the simulation if the source has a non-uniform radiation pattern. This can be controlled by means of the source's emission factor.

Warning! With object speeds that are 'too high' - specifically those approaching the speed of sound - the computation may get somewhat un-physical.

Tutorial: Table of Contents

This tutorial introduces you to the usage of the program and demonstrates some of its capabilities.

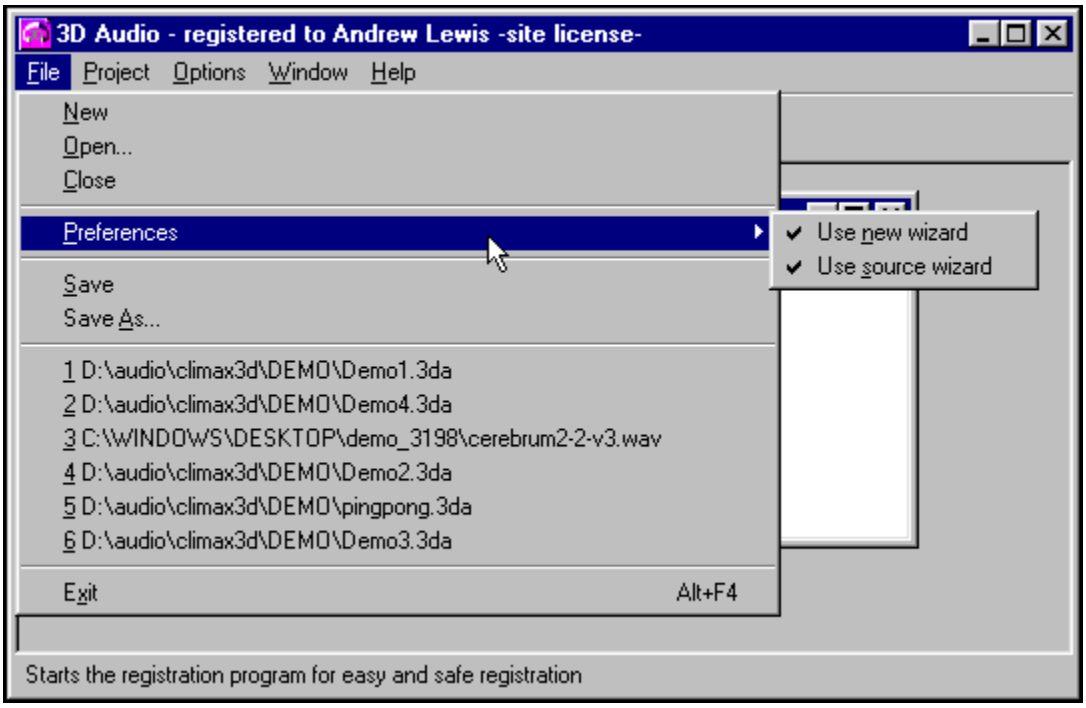
Two PCM Files are shipped along with this program, called "ping.pcm" and "pong.pcm". Both are the recording of a single bounce of a table-tennis ball. You might want to use other files for the tutorial, especially since more complex sounds tend to result in a more realistic rendered result.

The tutorial consists of the following sections:

Each contains a list of step-by-step instructions to follow, along with images to help visualize the interface. The resulting scene and project file is contained in the directory of 3D-Audio as project "demo?.3da", with the question mark indicating the section index number (1-4).

Creating a New Scene with the Wizards

1. Make sure that in the submenu **File / Preferences** both entries **Use New Wizard** and **Use Source Wizard** are checked.



2. Create a new project by selecting **New** in the File Menu or by pressing the button on the taskbar depicting a shiny, blank page. The first wizard will appear, guiding you through the various windows that set up the scene's basic parameters. In so doing, this **New Wizard** will solicit values for the generic, room and head variables - and by way of example you may enter the following data:

   a) **Output File**: The name of the PCM File to be rendered. Enter "output1.pcm" (no quotes).

   b) **Room Size**: The dimension of the virtual room in meters. Choose a 6 x 5 x 3 m room, i.e. 6 meters in x-direction, 5 in y- and 3 in z-direction.

   c) **Reflection Coefficients**: The amount of sound reflected by the flat surfaces of the virtual room. For purposes of this tutorial, enter 0.7 for the walls, 0.9 for the ceiling and 0.4 for the floor.

   d) **Position and Orientation of the Head**: First, position the head in the middle of the room at standing height; that is, choose the position (3,2.5,1.8). Leave the orientation as is, which yields a head looking along the x-axis.

   Next, the **Source Wizard** will appear and assist you in creating one source:

   e) **Source Name**: Enter a name to identify the source. Choose "Ball"

   f) **Source File**: The name of the PCM-File to associate with this source, i.e. define the sound that this source will play. For purposes of tutorial, you may

use the file "ping.pcm" within the 3D-Audio program directory. Click the "**Browse**" button and select it in the file dialog that appears.

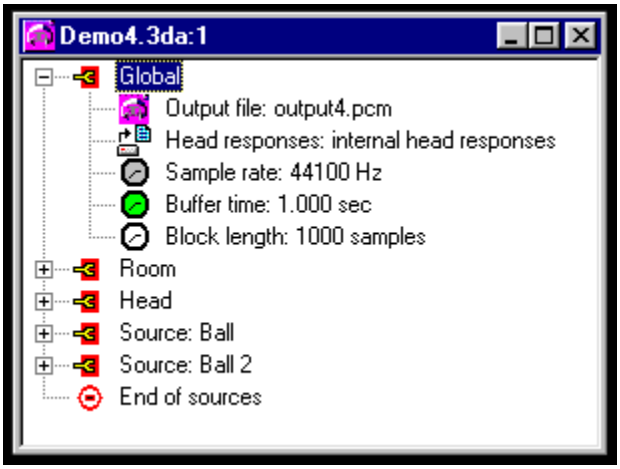
[One could also directly enter the file's path - however, this method will not automatically set the duration of the object to match that of the source file.]

PLEASE NOTE that an absolute pathname is needed for source files that are not located in the same directory as the project! This is because the wizard has yet to store the new project's database of input-file locations. As such, it has not needed to consider the contents of any directories other than its own.

g) **Position and Orientation of the Source**: The position of the source within the virtual room. The orientation will automatically be chosen such that it will point towards the head. Let us choose (4,4.5,0.1) as position. The source thus will appear to the left of the the virtual head near to the floor.

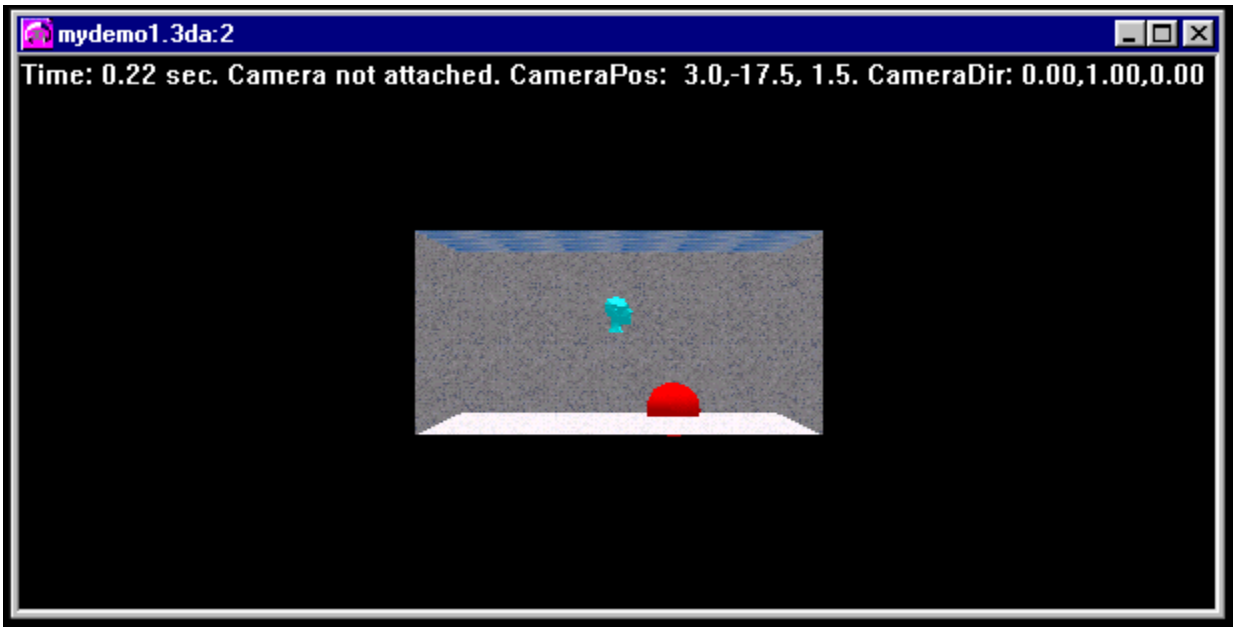
3. The wizard's setup is now complete. A window will open containing the project data in hierarchial structure, very much like directory tree. The project's parameters may be modified and appended by delving into this structure and editing the entries.

Save the created project as "mydemo1". It should be identical to the shipped project file "demo1.3da".



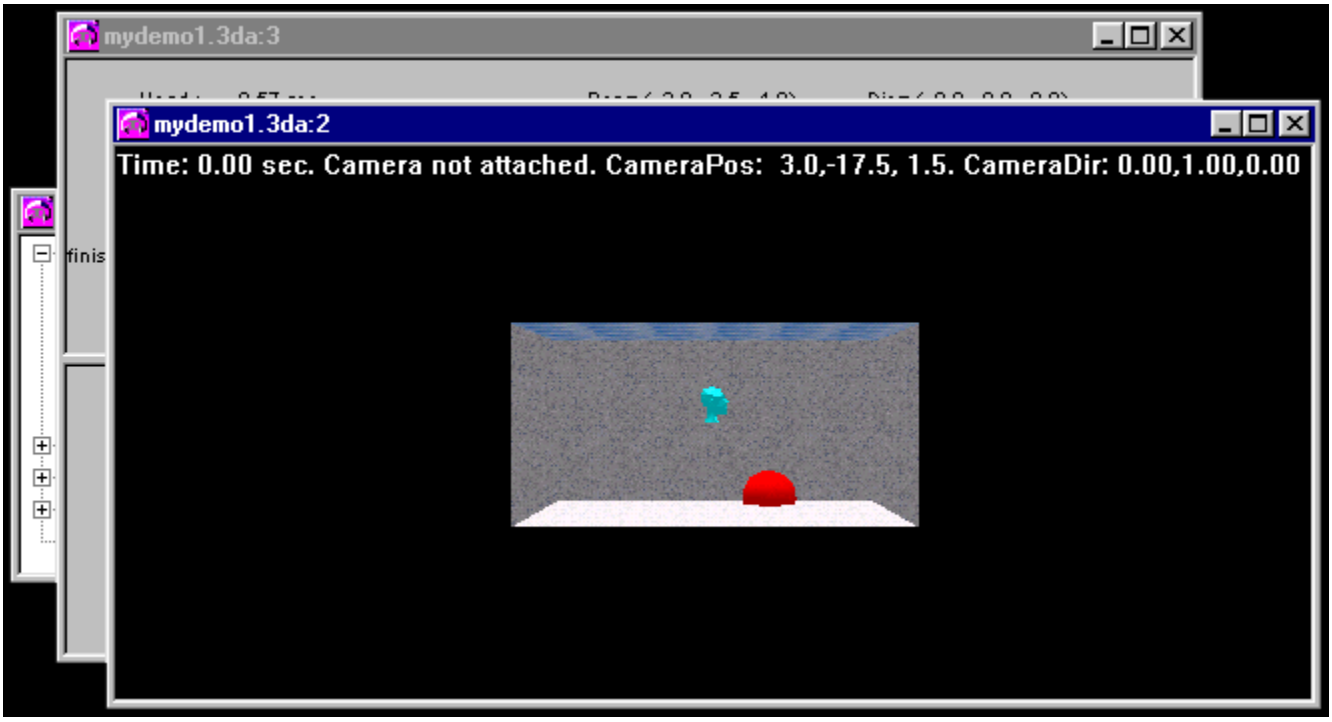
4. If DirectX3 is installed on your system, check the menu Options if it is selected for the visual display. If it is not, select "**Try to use Direct3D**"; the program will check if it can find it. If it can, you may need to restart the program now to allow proper use of the DirectX3-DLLs and subsequently reload the just created project.

5. Preview the scene by selecting **Project / Simulate**. A window will appear to visualise the scene. This preview is mainly useful to check animated scenes; since we have no animation yet, there is not much to see. If you have chosen "ping.pcm" as source input file, the animation will be over VERY quickly. This is due to the fact that the source file is only about half a second long.



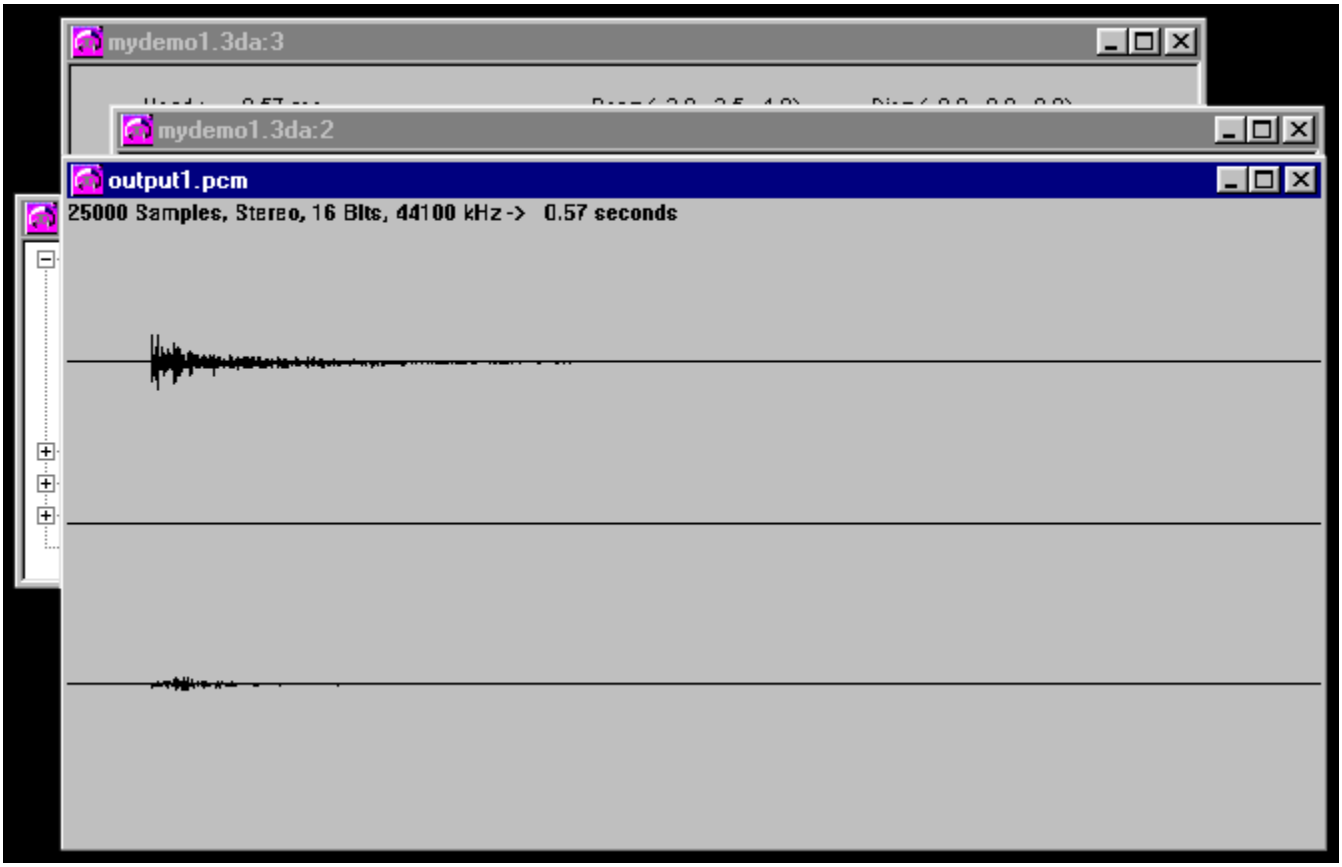
6. Finally, render the sound by selecting **Project / Compute**. Another preview window will be opened and updated during calculation, as well as an additional window containing progress data.

One may switch between these windows as desired, and even edit in the project window during computation!



This demo, however, is VERY short - so that calculation should be finished within a few seconds at maximum. After completion, you may open the rendered output file, in this case "output.pcm", by selecting **Project / Open Output File** or pressing

the seventh button from the left on the taskbar. This may also be done while computation is in progress. Upon completion, another window with the time-domain waveform of the rendered sound will appear.



With that window selected, one may listen to the result of the calculation by pressing the play button in the taskbar, or by selecting the menu item **Audio / Play**. Don't forget that you will need headphones to achieve the desired result! Since the sound is quite short, select **Audio / Loop** (or the second button from the right in the taskbar).

If playback is initiated, it will loop until halted by **Audio / Stop**. The spatial effect should be perceptible, if not very impressive. This is because the computation time was kept to a minimum, so in the next section we'll advance the project...

7. To continue, please close all windows (except the project window) and click here.

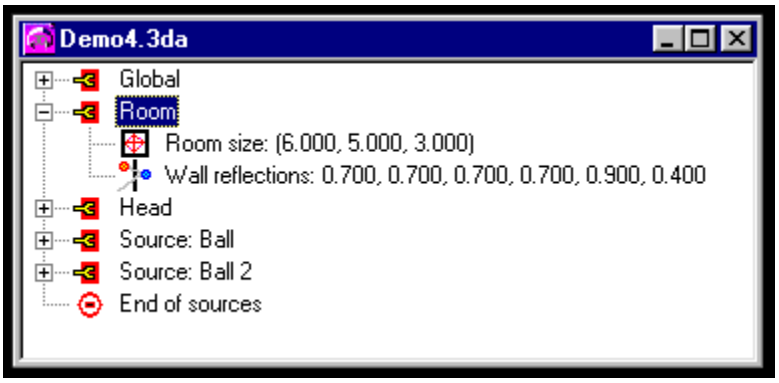
Changing Parameters of the Sound Source

Listening to the result of the first calculation, you will find that

- The source sound is too short
- The spatial information carried by the sound is not as impressive as you had hoped

To mitigate these drawbacks, do the following:

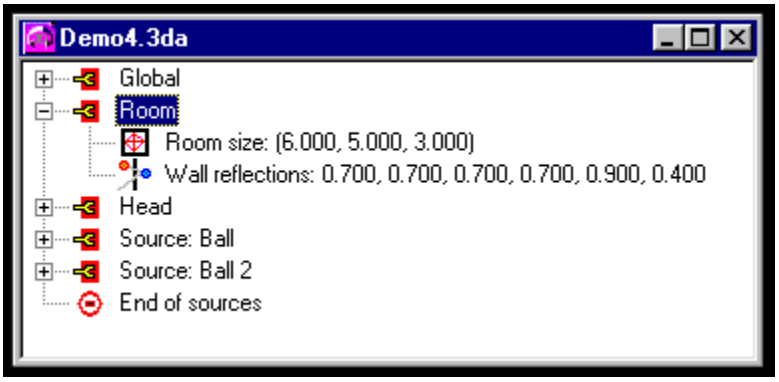
1. In the Project window, double-click the branch marked "**Global**" at the very top of the window. The tree-like structure will open, displaying several parameters that are owned by "**Global**". To avoid over-writing the results from section one, we must now specify a new name for the output file.



Double-click the entry "**Output File**"; a parameter window will open, containing a number of notebook-tabs. These tabs correspond to the various entries, as displayed in the project window.

Change the name of the output file to "output2.pcm" and close the window. Double-click the top-level entry "Basic" once more and the structure will collapse, hiding the subitems once again.

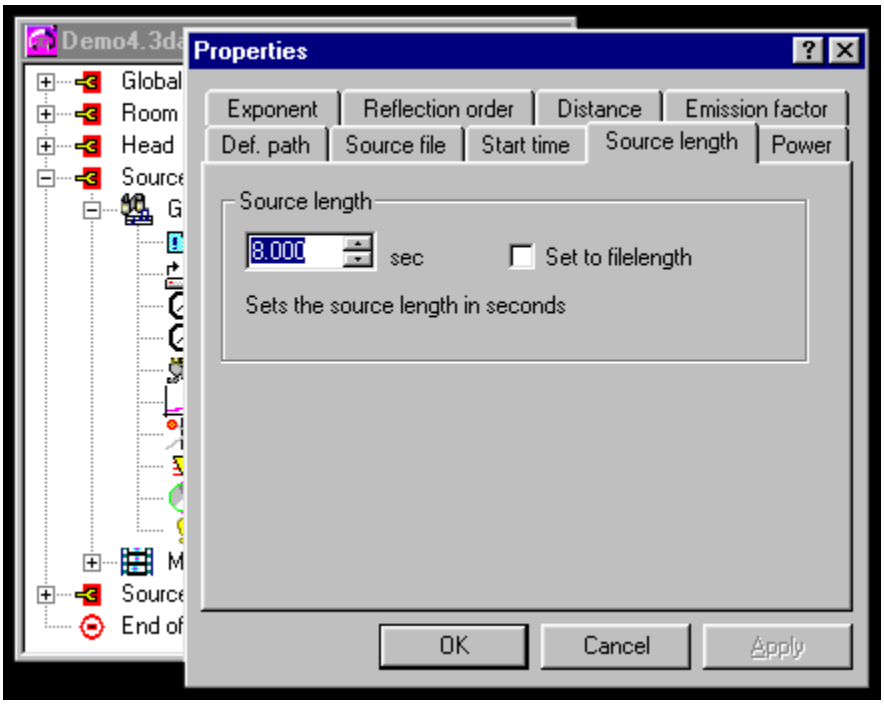
2. Now double-click on the entry "**Source: Ball**". The structure will open, revealing two subitems of the source entry: **Global** and **Motion**.



The first contains further entries concerning the basic setup of the source; the latter contains entries that define the motion of the source within the virtual room. We will now concentrate on the generic parameters; to access them, double-click on

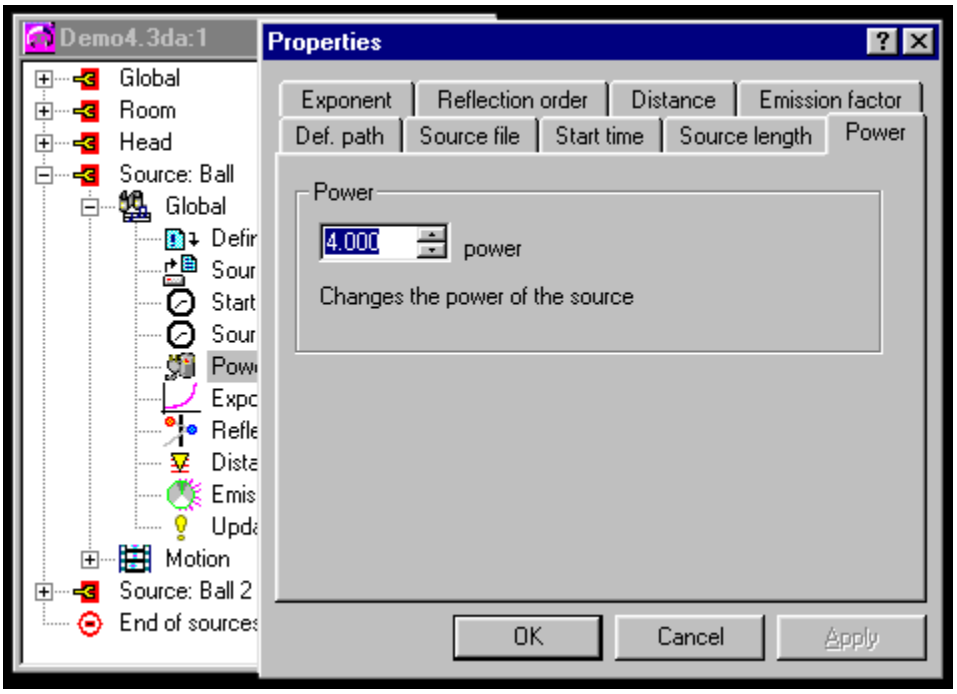
the subitem "**Global**". Various entries will appear, defining the behavior of the source.

First, we want the sound of the bouncing ball to be repeated. To achieve this, double-click the entry "**Source Length**". A parameter window will open, displaying a "Source Length" tab along with a number of others. [These tabs correspond to the various entries displayed in the project window.]



Set the length of the source to 3 seconds, and close the dialog. If the length of a source is set to be longer than its input file, the sound will be looped until it fits.

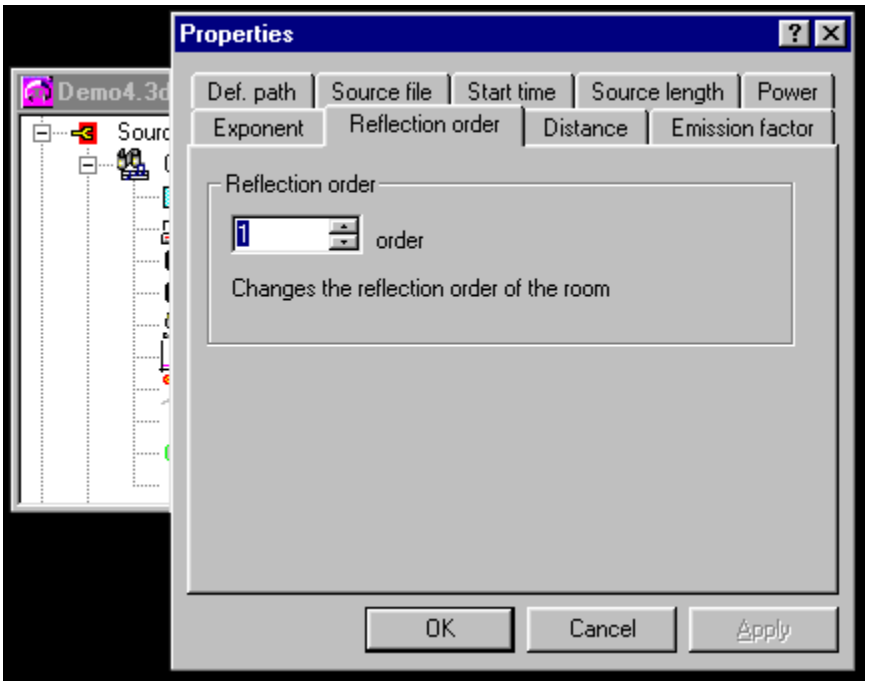
3. Now select the entry "**Power**" in the project window hierarchy. The same window will appear once again, showing a different tab.



Set the power to a value of four to increase the volume of the sound. This power is an absolute factor that the source waveform will be amplified with.

When using the wizard to create a new source, the power will automatically be set to a value which yields a medium volume of the source. This initial power depends on the distance from the virtual head to the source, as well as upon the path-loss exponent.

4. Finally, select the entry "**Reflection Order**". This value defines the maximum number of reflections, N, that a sound emitted from the source will experience.



Waves resulting from more reflections will not be considered for calculation. The more reflections used, the more natural the result will be. The computation time, however, increases exponentially with N!

An order of zero means that only the direct line-of-sight wave from the source will be accounted for. The default value for this is one, which yields a reasonable tradeoff between quality and computation-time. If you have set up a scene and wish more naturality, you can later increase this value.

PLEASE NOTE that additional reflections of higher order change the perceived power of the signals! Set the value to two for now. Click here to find an overview of the number of reflection beams corresponding to a certain reflection order.

5. Save the project as "mydemo2". The shipped project "demo2.3da" should contain the same.

6. Redo calculation by selecting **Project / Compute**. This time the computation will take much longer than in the first section. This is due to the fact that we utilize reflections of higher order, and that we extended the length of the source to three seconds.

7. Open the rendered result by means of **Project / Open Output File** and listen to it. You will find that the spatial impression has been slightly improved; if you use source files different than "ping.pcm", you might find them more compelling.

Using more reflections is definitively a good idea in terms of resultant quality. If one has the time, try reflection orders as high as five to seven [seven is the absolute maximum to finish calculation in finite time]. An order of seven requires a calculation time of several thousand times the length of the source waveform; as such, rendering ten seconds of sound can easily take a whole night.

DO NOT, however, use the shipped table-tennis ball sounds for that experiment,

since the results won't be too impressive. Perhaps try some speech or a melodic instrument instead.

8. To go on, close all windows except the project window and click here.

Animating the Sound Source

We now will improve the spatial impression by introducing motion. This greatly enhances the spatial perception because it enables your brain to additionally evaluate the change in the perception for estimating position. We will make use of different types of motion to demonstrate the use of the supported motion instructions. Perform the following steps:

1. Close all windows except the project window when continuing from section two, or reload the project from section two (either "mydemo2.3da" or the shipped "demo2.3da").

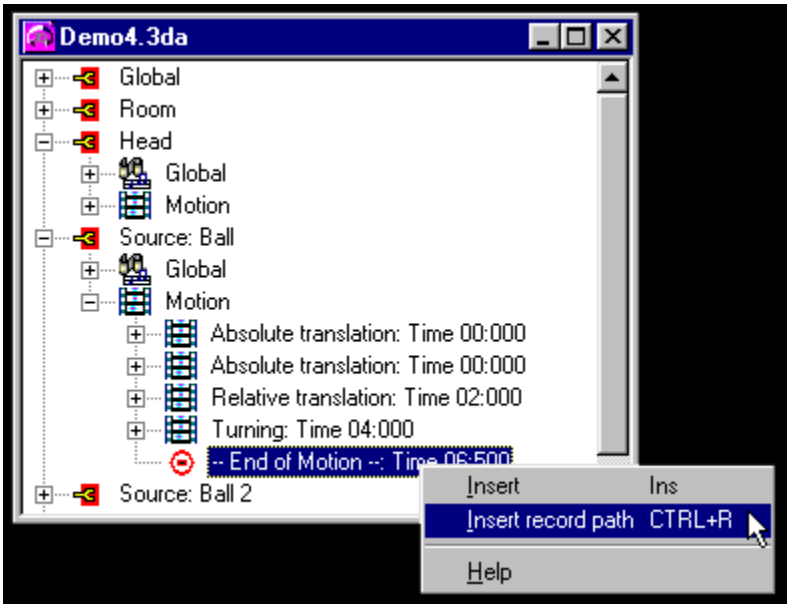
2. Change the name of the output file to "output3.pcm" as described in section two, step one.

3. Change the reflection order of the source back to one [to save time]. See the last section, step four. Set the length of the source to eight seconds.

4. Open the contents of the source and subsequently open the contents of the motion entry: first double-click on "**Source : Ball**" and then upon "**Motion**". The motion entry will contain a list of items [instructions] that define the source's motion path during simulation. Presently, the list will only contain one entry - "Absolute Translation".

   This motion instruction is used to place the source at the desired initial position, as specified within the **Source Wizard** [see section one]. You can now add further instructions to the list to define a desired motion sequence. To add an instruction, select on of the entries of the motion list and press the right mouse-button.

   The context menu that appears will allow you to edit the selected item [menu item "**Properties**"], delete it or to insert a new one. Inserted items will be placed before the current item in the list. To append an item, select the one below it, right-click and choose "**Insert**". To add an item to the end of the list, select "**-- End of Motion --**" and then Insert.



We now will append a number of instructions to the list and edit the paramaters of each one.

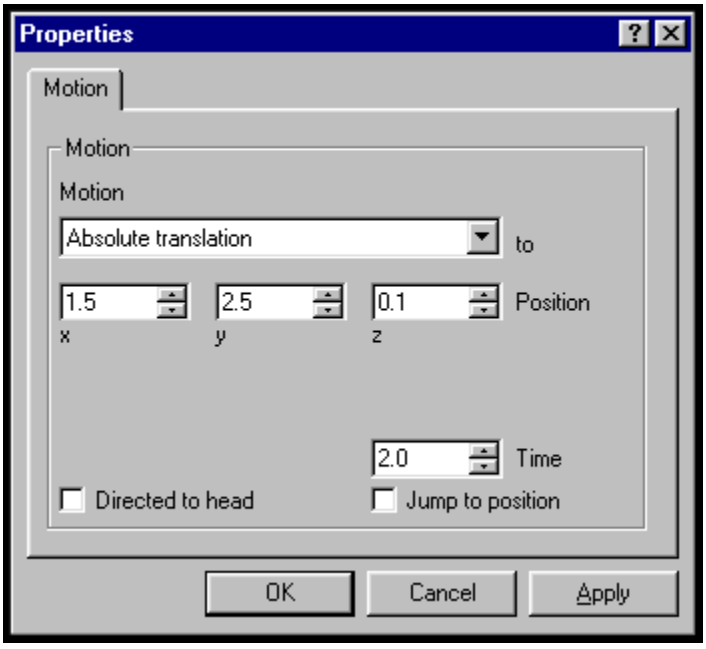
First, select the "End of [List]" entry, press the right mouse-button and select "**Insert**" from the context menu. The new instruction will be placed at the end of the list and will be highlighted. Next, select "**Properties**" from its context menu (remember that double-clicking opens up a branch to reveal its sub-items).

A window will open, containing the parameters of that motion instruction. Because the parameter happens to be an "Absolute Translation", the Motion class will be set to that type, along with the specific options that pertain to it.

In this case, they are the X-Y-Z coordinates, the ending time for the movement and two checkboxes - if the source should maintain its relative orientation towards the virtual head, and whether the movement should take place instantaneously.

[The orientation of a source is only of interest when using directed-emission characteristics for the source or walls; as such, we will not be make use of that feature in this tutorial. You may, however, change the emission characteristic within the "Global" entry of the source in the project window.]

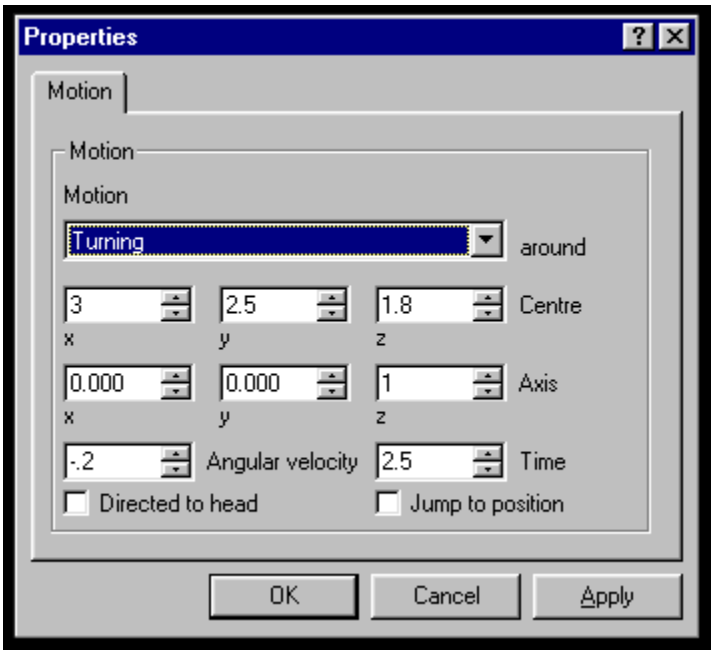
a) Keep the first instruction as "Absolute Translation" (this is the default) and select the end-position as (1.5,4.5,0.1) to be reached in two seconds. Close the dialog.



b) Repeat the step described above to append another instruction to the list and invoke its parameter window. Set the type of the motion to "Relative Translation" with a velocity of (0,-1,0) and a time of two seconds. This will lead to a position of the source after the motion at (1.5,2.5,0.1). Close the dialog.

c) Append another motion instruction and edit its properties. Set the type to "Turning", choose the centre to be the position of the head (3,2.5,1.8) and the

axis along the z-axis (0,0,1). Set the angular velocity to -0.2 - the sign gives the direction of the motion - and the motion time to 2.5 seconds.



This causes the source to move on a circular path around the head, with a speed of 0.2 revolutions per second for three seconds. Hence the source will describe approximately a half-circle. After this motion the source will come to rest in front of the virtual head.

5. Save the project as "mydemo3". It should be identical to shipped "demo3.3da".

6. Choose **Project / Simulate** to preview the motion. Click the mouse within the rendering window that opens, and drag the mouse-cursor to change your viewpoint.

Click here to learn more about the use of keys within the rendering window to change the display and viewpoint.

7. Compute the scene. Check the output during calculation. When listening to the result, the direction of the sound source is clearly perceptible.

   Please also note that the settings within the tutorial are not chosen with respect to maximum listening enjoyment, but rather to be instructive.

   One may repeat the computation with higher reflection orders - but as already mentioned in the last section, this bouncing sound is not very well suited for higher order reflections. Try substituting the source file with a speech file instead.

8. To go on, close all windows except the project window and click here.

Motion Instructions

The following motion instructions may also be applied to the virtual head:

Absolute Translation: Moves the source to an absolute position within the virtual room. Leaves the orientation of the source unchanged, unless "**Directed to head**" is checked (see above).

Relative Translation: Moves the source with a given speed and acceleration. Leaves the orientation of the source unchanged, unless "**Directed to head**" is checked (see above).

Absolute Rotation: Rotates the orientation of the source to an absolute angular value. Leaves the position of the source unchanged.

Relative Rotation: Rotates the orientation of the source with a given angular speed and acceleration. Leaves the position of the source unchanged.

Turning: Moves the source on a circular path around a given center and axis. Leaves the orientation of the source unchanged, unless "**Directed to head**" is checked (see above).

Adding More Sources

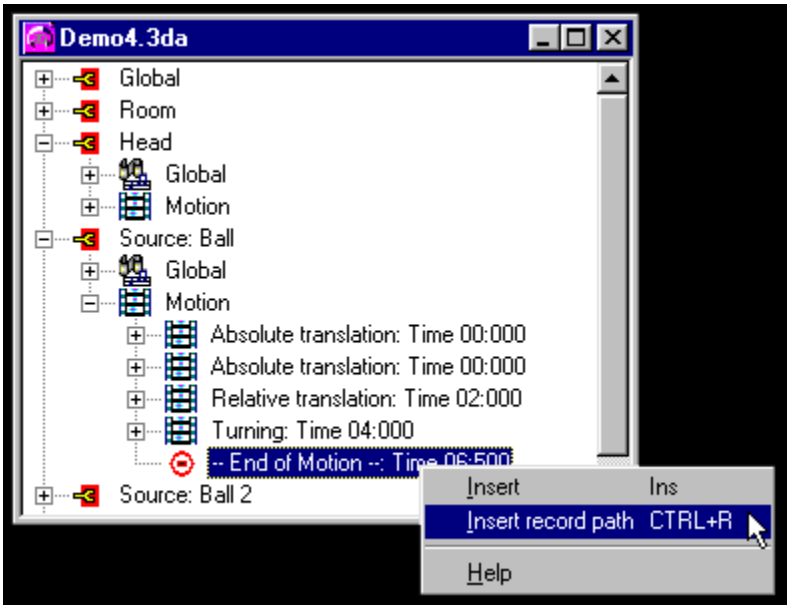
Now we shall expand the scene to include more than one sound source - by adding another one with the **Source Wizard**. To do so, please follow these steps:

1. Revert the status of the project back to that which was saved in the last section, e.g. load the supplied "demo3.3da".

2. Set the name of the output file to "output4.pcm".

3. One may add new sources within the Project window in very much the same method described in step four of the last section. [Select the "End of sources" entry and press the right mouse-button.]

   Next, select "**Insert**" in the context menu.



A new source will be added and the Source Wizard will appear, guiding you through the setup or the source's basic parameters. If the Wizard is disabled (under **File / Preferences**), all settings will have to be inserted using the Project window.

For now, we'll assume the Wizard to be active:

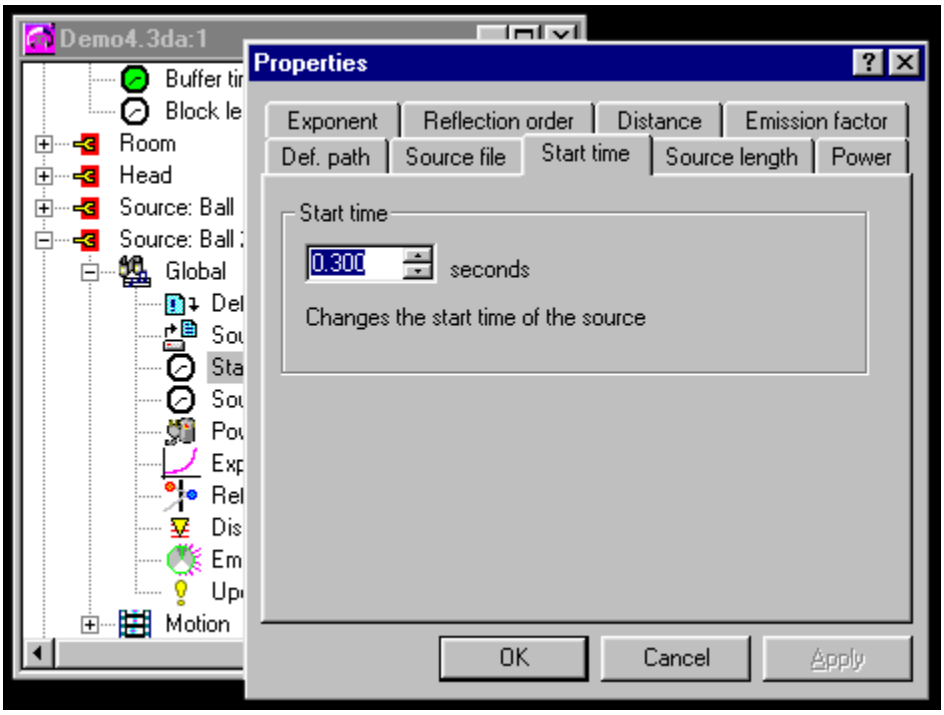
e) **Source name**: Enter a name to identify the new source. Could be "Ball 2"

f) **Source file**: The name of the PCM-File to associate with this source, i.e. define the sound that this source will play. Use the file "pong.pcm" within the 3D-Audio program directory. This sound differs slightly from "ping.pcm".

g) **Position and orientation of the source**: The position of the source within the virtual room. The orientation will automatically be chosen such that it will point towards the head. Let us choose (2,1.5,0.1) as position. The source thus will appear behind the virtual head, to the right and near to the floor.

4. Within the project window, use the hierarchial display to find the entry "**Start Time**"

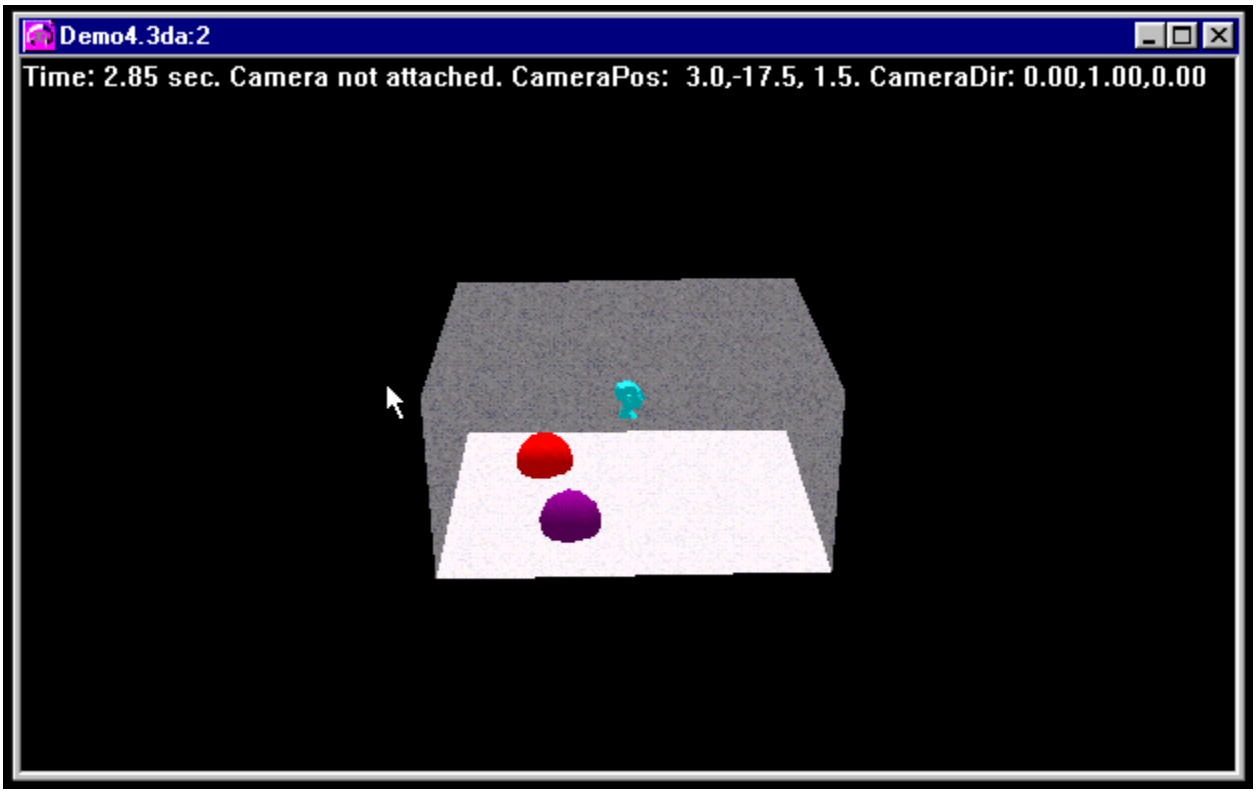
which resides in "**Global**" within the new source "**Source: Ball 2**". Change the value to 0.3 seconds.



This causes the new source to start after 0.3 seconds, so that there will be a little offset between the two bouncing sounds. Additionally, change the length of the source sound to eight seconds.

5. The newly created source is not animated yet; you should experiment with this, to practise the usage of the program. SAVE THE PROJECT OFTEN!

6. Choose **Project / Simulate** to view the altered scenery.



7. Compute the result and listen to it; the two sources should be easily distinguishable.

NT 4.0 Technical Note

There seems to be a problem with the screensaver and the Direct3D output in Windows NT 4.0. If you've installed Service Pack 3, make sure to turn off the screensaver.

If you do not turn it off, **3D Audio** will lock when the screensaver appears!

Distribution Status

You may freely distribute the unregistered version of 3D-Audio, provided that all the files are included and are unmodified and that no files have been added to the package. Please distribute it by copying the original .ZIP file. You may not accept any money for the distribution.

If you'd like to include this software on a freeware/shareware CD-ROM or other compilation, please contact CSS before doing so, to be sure that you are not including old or incomplete stuff in the compilation. We would also appreciate a free copy of the CD-ROM!

KAGI Registration Service

Using Kagi registration service offers these advantages:

- Secure SSL internet online registration via KAGI for users with a credit card. **Go to the SSL online registration** or if your browser does not support SSL try this site!
- Virtually any payment method is accepted: Cash, Check, Money order, Credit cards, Invoice.
- Credit card numbers will only appear encrypted within the registration order, increasing the security of the transaction.
- You can submit the order via e-mail, fax of mail.
- Kagi is located in USA, so it is best for all users out of Europe. Residents in Europe should pay directly to us. This will speed up the registration process!

Click here to start the KAGI registration program.

Revision History

V1.2 02.05.1998

- Have a look at the brand new helpfile: re-written, re-edited and additional documentation for 3D-Audio by **Andrew Lewis**. Thank you very much for your great work!
- Some small bugfixes

V1.1c 02.02.1998

- One may choose the output of the audio-wave device now .
- Minor bugfixes, including the registry problem with Win95-OSR2.

V1.1a 19.11.1997

- Bugfix: 'opening an invalid output filename' will now be handled correctly!

V1.1 15.11.1997

- Bugfix: opening, closing and re-opening a project will no longer crash 3D Audio when visualising with Direct3D.

V1.1 14.11.1997

- Included capability for Motion Capture by means of providing interface functions for plugin DLLs!
  - Simple recording plugin DLL for input via joystick, mouse and keyboard
  - Free SDK for implementing custom DLLs
- Miscellaneous bugfixes

For registered users only:

There are two more sets of head-responses freely available:

- The 'diffuse set': the head responses, which are manupulated by a small reverb effect
- The original set: these are the original recorded responses, which provide the best quality possible.

Registered users also get a program (HRTFconverter) to convert the public HRTF sets from the MIT originals for free! Convert the data from the MIT (ftp://sound.media.mit.edu/pub/Data/KEMAR) into a valid 3D Audio HRTF file! You can manipulate the original data [e.g. with a reverb] and then convert the impulse responses into a valid 3D Audio HRTF format!

If you want to get one of these utilities, contact CSS! They are free for registered users!

V1.0a 09.09.1997

- Registered users can get two more sets of head-responses for free:
  - The diffuse set: the head responses are manupulated by a small reverb effect
  - The original set: these are the original recorded ony small truncated head responses for best audio results.

- Slight improvment of the rendering algorithm
- DirectX3 upgrade bug solved
- Export as .WAV format
- Playing .WAVs no longer crashes the program
- Miscellaneous bugfixes
- Extended help
- **Secure SSL internet Registration via KAGI**

Official release V1.0 25.7.1997

- New render algorithm; about 300% faster than before
- Support for wav-files as sources
- More keyboard handling for faster processing
- Extended help
    - Smoother thread shutdown
    - Several bugfixes

V0.99b 19.6.1997

- Fewer shareware restrictions: the program will not stop working after 7 days!
- 'New Wizard' and 'Source Wizard' added to simplify the setting up of basic parameters for
    the head and source(s)
- Help extended
- Demonstration tutorial with four demo setups added
- Several small bugfixes

V0.99 8.6.1997

- Initial release

Planned Features

Here are some of the planned features for the next versions of **3D Audio**. If you have an idea for a cool feature, please contact us!!

Version 1.5

Surround Sound Rendering

- Rendering of surround sound with unlimited numbers of fixed boxes in the virtual room. The output file(s) will be a n-channel file where each channel represents the output of one box. Just animate the sound sources and hear them spinning around!

Version 2.0

Audio Format Compatibility

- Converter between different sound formats, e.g. stereo<->mono
- sample rate converter

3D Rendering

- Improved renderer
- Direct support for .WAV export
- Audio - preview with DirectSound3D [ut this will sound different from the renderer output... as yet an unresolved problem]
- Server for DirectX /ActiveMovie streaming [see below]

Scene Construction

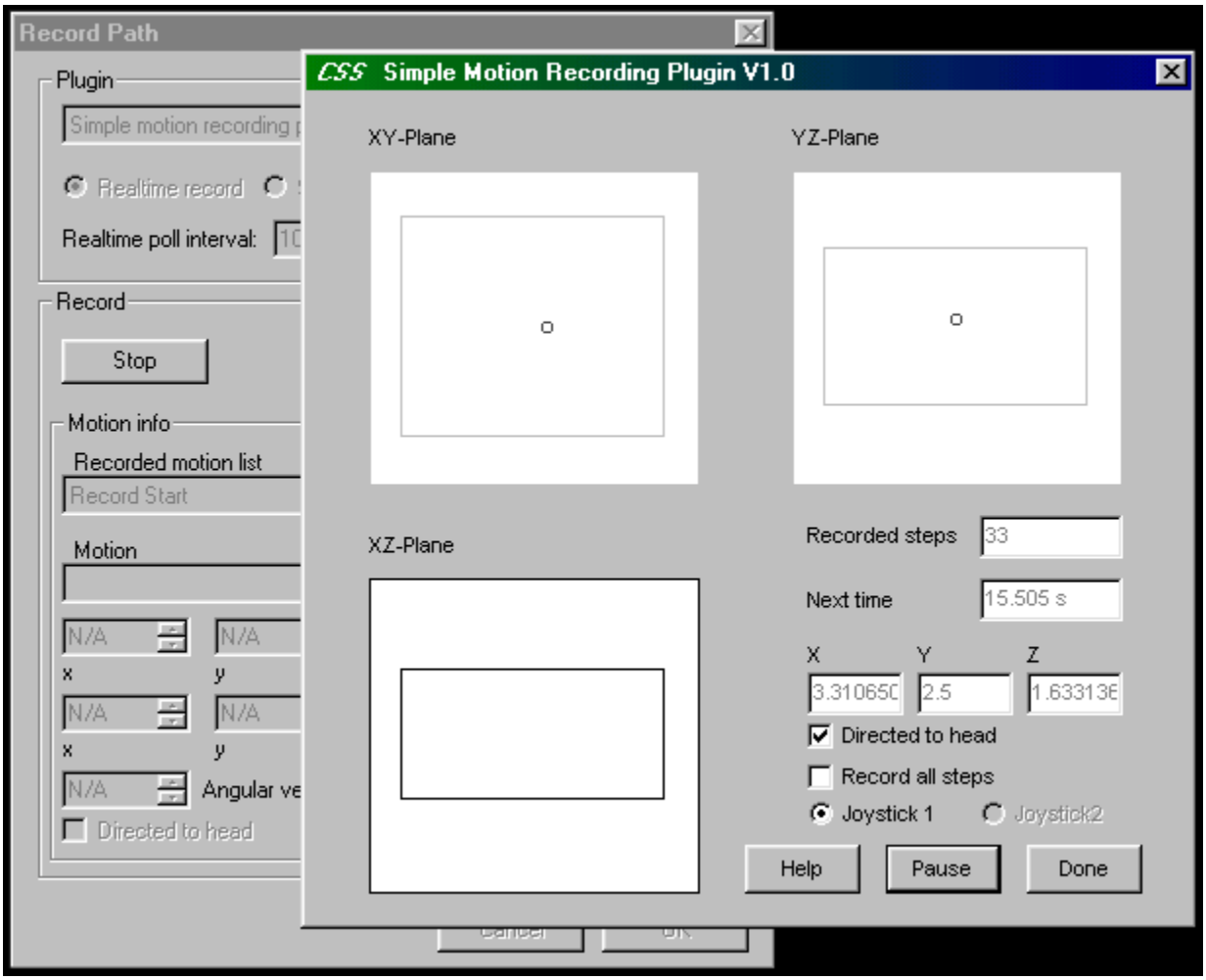
- Movement construction within the preview window
- More flexible movement paths,like splines et cetera.
- Arranging Timeline

As a Plug-in for Other Programs

- **3D Audio** will become a DirectX-streaming client for use as a plug-in compatible with participating applications such as Sound Forge, Cakewalk, Cool Edit, and so forth.

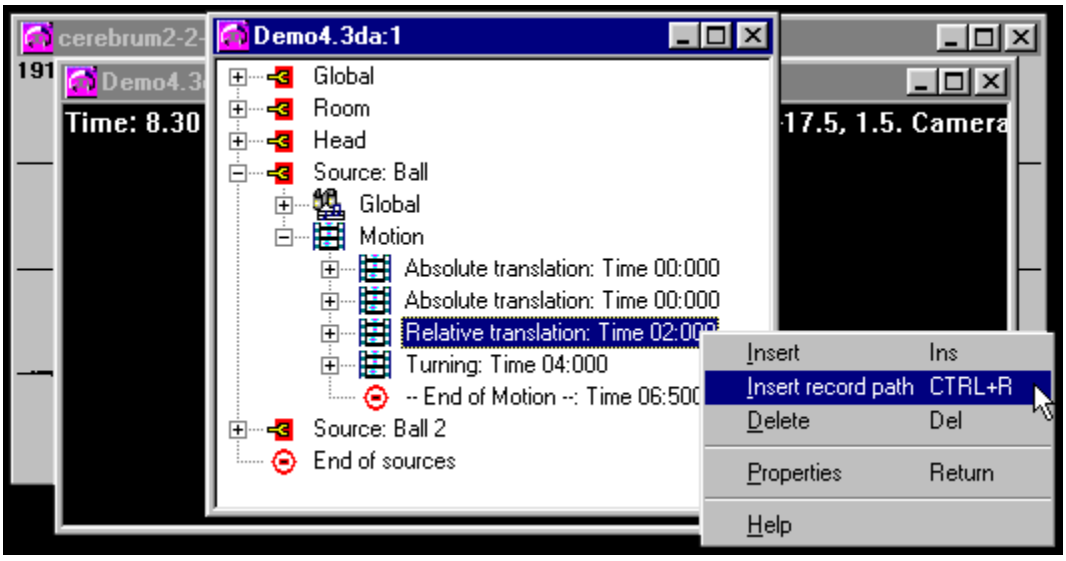
Motion Capture Plug-ins and 3D Audio

Versions 1.1 and later of **3D Audio** allow one to prescribe motion paths for sources and/or the virtual head. These paths are recorded by motion capture, and are described either in real-time or in steps.



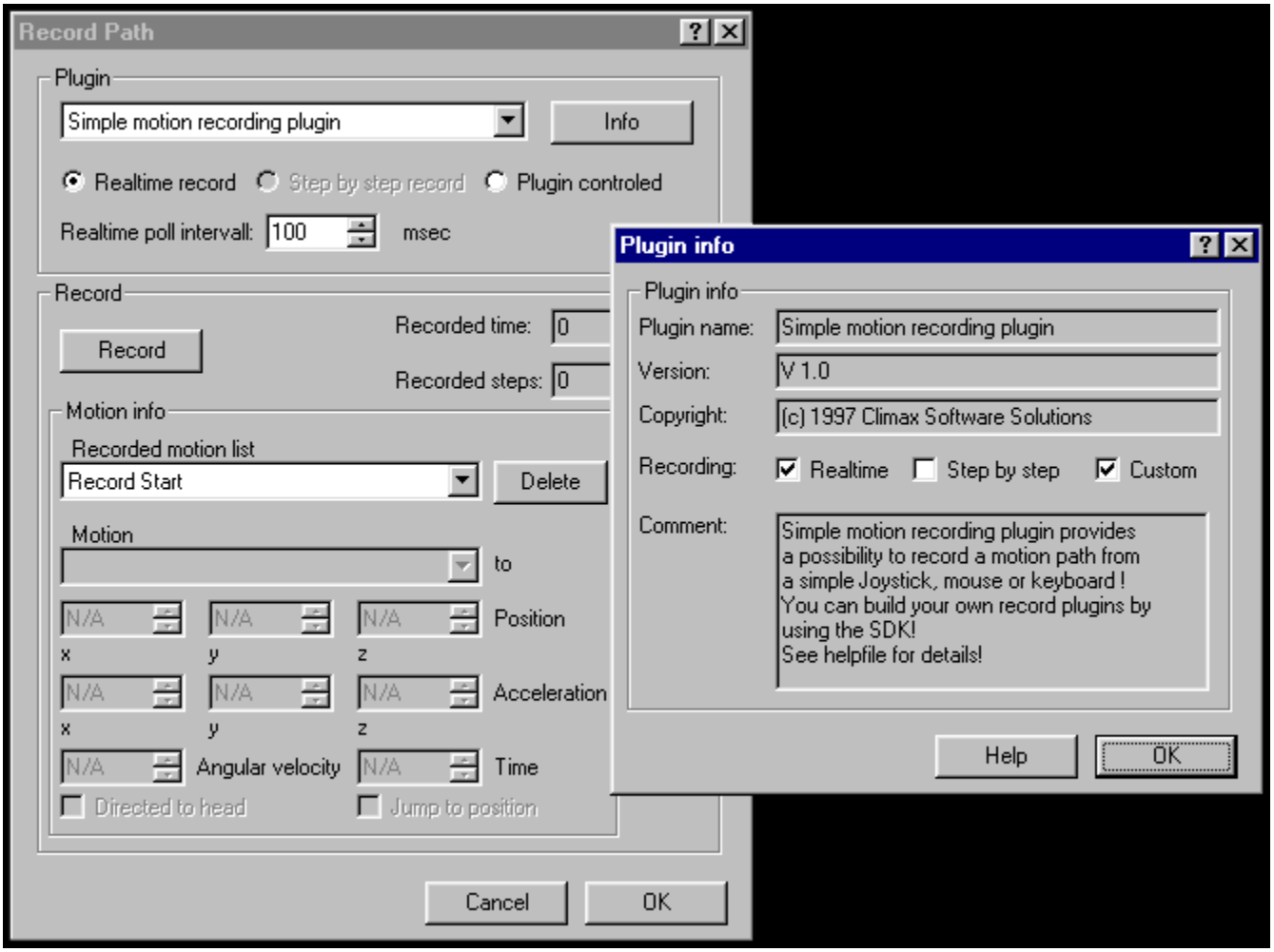
PLEASE NOTE that one will get the menu for recording motion on valid branches only. A "valid branch" is defined as being either a Head or Source motion.

In addition, jumping from the base of the tree to a specific location in a motion command-list requires first a left-click on the individual command entry [to select it] and then a right-click to access '**Insert Record Path**' in the associated context menu.



Alternately, one may press CTRL+R after selecting a valid entry.

The plug-ins can support one or more of three available recording modes:

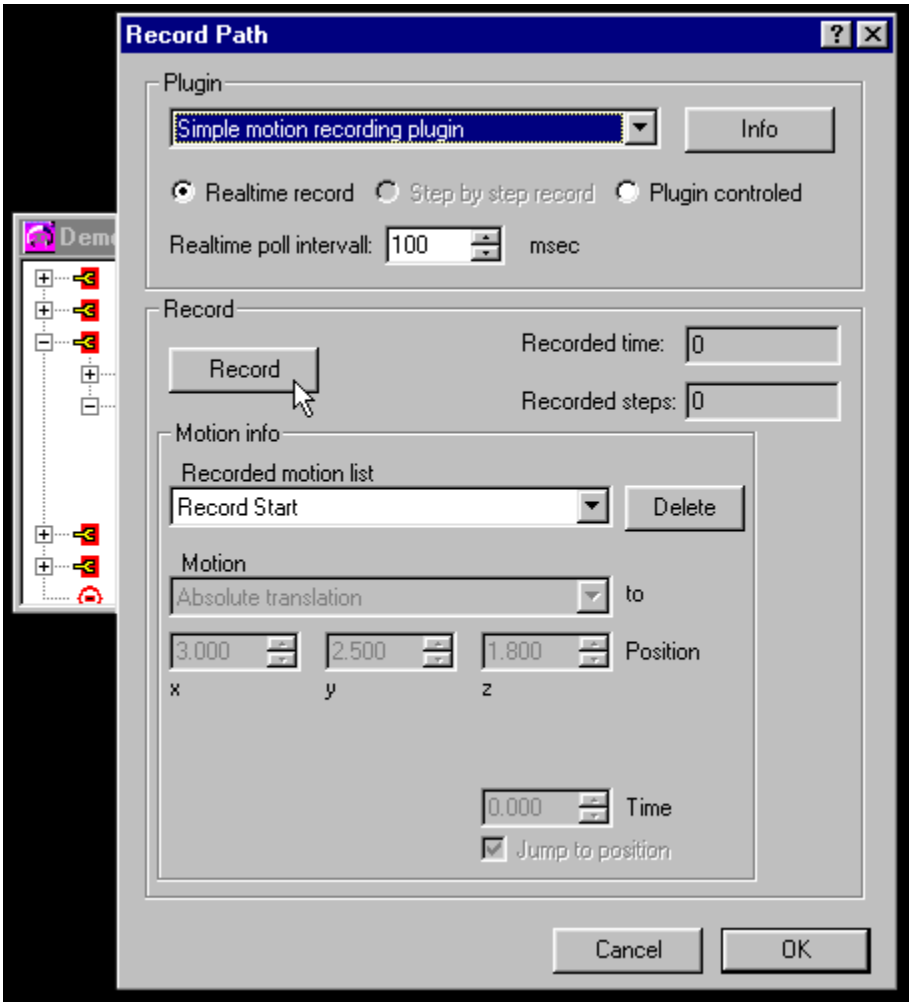


1. **Realtime**: The plug-in is constantly polled for new position data, with **3D Audio** handling the time scale;

2. **Step-by-Step**: The plug-in is polled for new position data, with the plug-in handling the time scale;

3. **Plugin-controlled**: The plug-in takes control of the recording process, and returns when recording is finished. Step recording is enabled with the 'Step by Step' button behind the 'Info' button; likewise, the 'Custom' checkbox activates that excellent feature in your mocap plug-in.

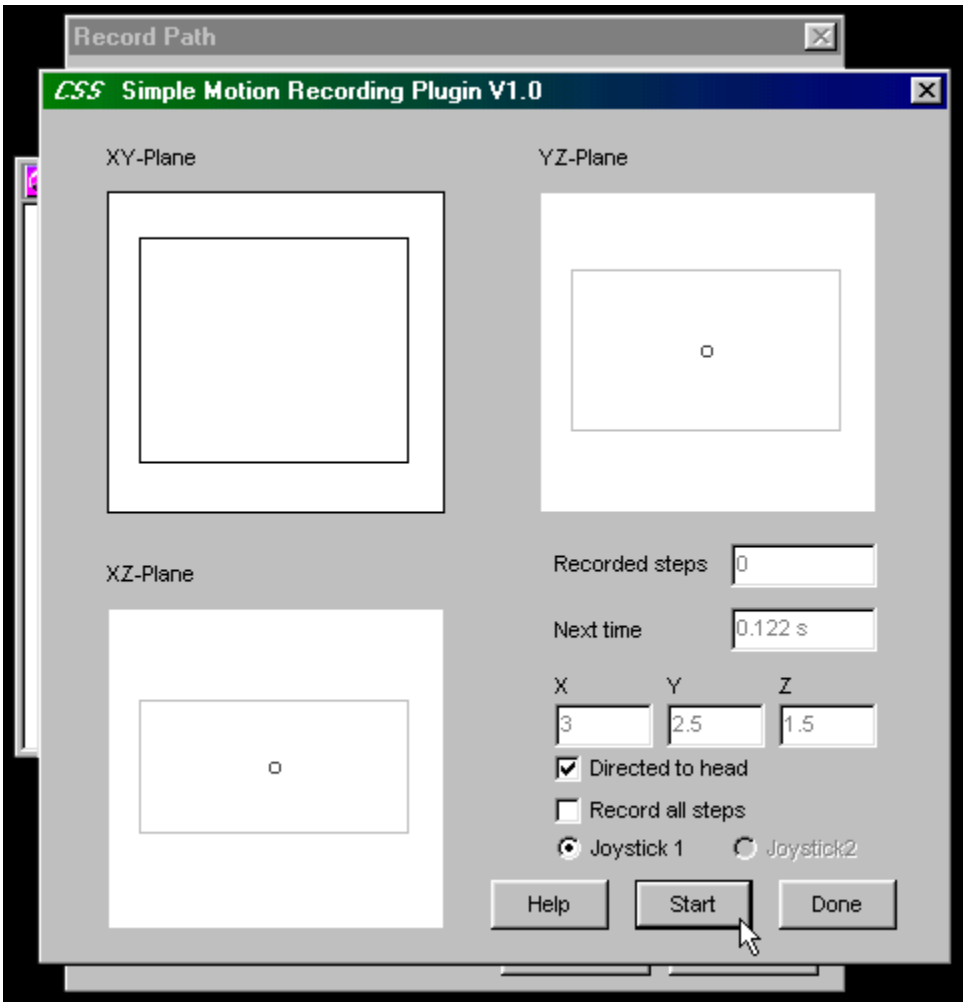
Once the options are set, click the 'Record' button to begin with the selected Plug-in, recording mode, and polling interval.



REAL-TIME CAPTURE

Click 'Start' in the **CSS Simple Motion Capture** dialog to begin actual motion capture. If the 'Real Time record' box is checked, capture will begin with the 'Start' button; however with real-time captures you are given the option to have the session data actually start at either 00:00:00 or the time of the first recorded motion.

The dialog indicates the object's current XYZ coordinates and a representation of the virtual room. In addition, it displays the current elapsed time and number of steps captured.

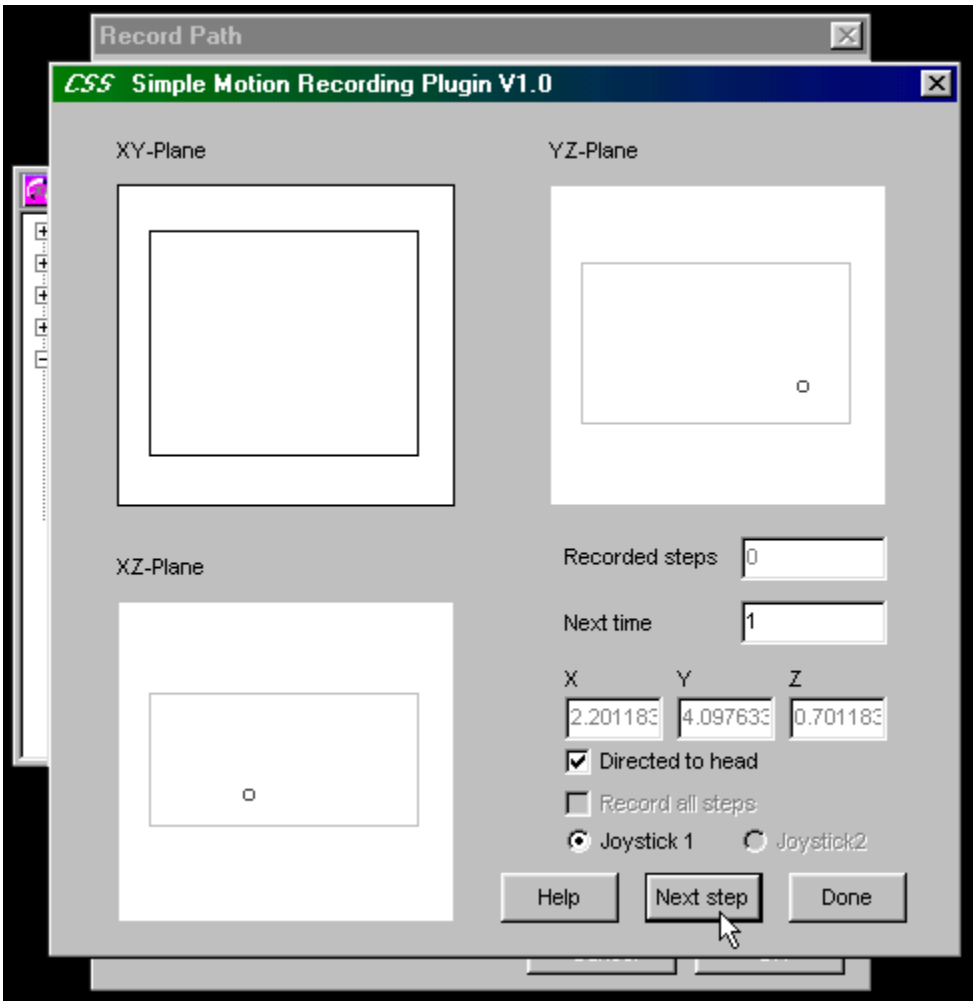


Click 'Start' to begin dragging the motion path around in the three plane views. The position of the target is visually updated in the other planes at the same rate as the polling interval. Recording may be suspended and resumed with the 'Pause' button, which is only visible during real-time recording. When finished, click 'Done'.

There is also a 'directed to head' checkbox, which automatically orients the captured source data towards the virtual head.

STEP CAPTURE

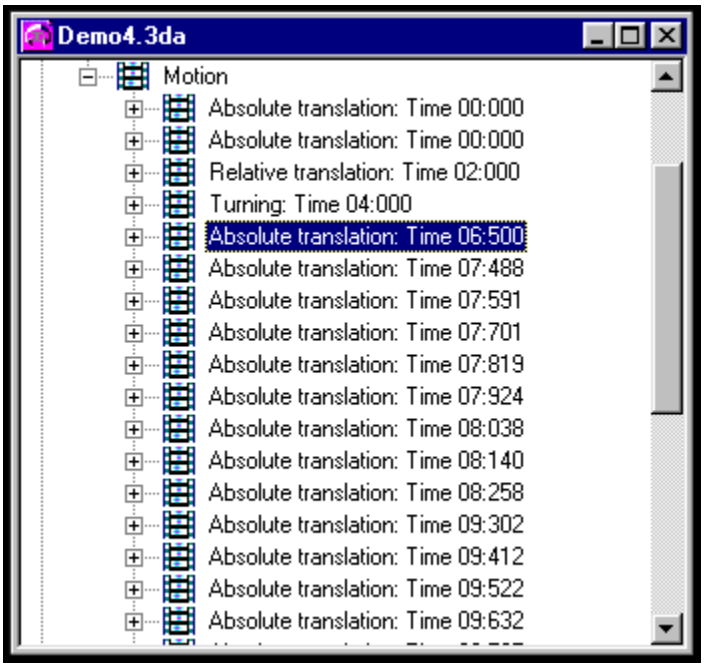
Recording in Step mode allows one to pinpoint the individual movements and their durations. Position the cursor for each instruction and click 'Next Step'.



Repeat the process until satisfied, and then click 'Done'.

The **Record Path** Dialog will now display the elapsed time and number of steps recorded, and the 'Delete All' button that appears will purge the session cache. If the 'Record' button is pressed again, the new instructions will be appended to those previously captured.

All commands recorded by the plug-in appear in its **Record Motion List**, which allows anything in the session to be reviewed, modified, or deleted. Hit 'OK' to approve the motion commands and insert them in the Project window. Or, choose 'Cancel' to abandon the plug-in and all data pertaining to the mocap session.



If the session is approved, each recorded instruction will appear in the Project window, on the tree under the Source / Motion branch, beginning at the insert location.

3dplugin.h

```
#ifndef _3DPLUGIN_H_INCLUDED
#define _3DPLUGIN_H_INCLUDED

// Defines for recording modes
#define RELATIMERECORD          1
#define STEPBYSTEPRECORD        2
#define PLUGINDEFINEDRECORD     4

#define MOTIONPLUGIN 1

// Defines for the types of motion
#define Plug_Translation_abs        1;
#define Plug_Translation_rel 2;
#define Plug_Rotation_abs   3;
#define Plug_Rotation_rel    4;
#define Plug_Turning                5;
#define Plug_TransRot_abs           7;

#ifndef __PluginImport__
#define EXPORT __declspec(dllexport)
#else
#define EXPORT __declspec(dllimport)
#endif

// Attention!!
// All structures must be compiled with BYTE alignment!!!

// Byte alignment in Borland:
#pragma pack(1)


// Struct holds info on plugin
struct M3DAudioPluginInfo
{
        char Name[256];             // insert Plugin Name
        char Version[256];          // insert Version Number
        char Copyright[256];        // insert Copyright information
        char Comment[1024];                 // insert short description information

        int Capabilities;                   // insert Capabilities (RELATIMERECORD, ...)

        int PluginIdentifier;       // identifies the Plugin, use MOTIONPLUGIN for valid DLL!
};

// Struct holds data of motion instruction
struct M3DAudioMotionParameter
{
        double MotionParameters1[3];    // First parameter vector
        double MotionParameters2[3];    // Second parameter vector

        BOOL    FixedToHead;        // TRUE if the source is always directed to the head
        double AngularVelocity;             // For turning motion
```

```
	int	Motion;				// Type of motion, see defines above

	DWORD TimeToEvent;			// Time that this motion takes in msec.
};

// Struct to hold additional info when recording realtime or step-by-step
struct M3DAudioMotionRecord
{
	M3DAudioMotionParameter Parameter;	// Motion instruction
	DWORD RecordTime;			// in msec since realtime recording started

	BOOL Changed;				// TRUE if position changed; else instruction will
be discarded
	BOOL Paused;				// TRUE if recording is paused when realtime
mode is running
};

// Struct with info to initialise the recording
struct M3DAudioPrepareRecordInfo
{
	HWND ParentWindow;				// Handle to parent window; use this to
open your window
	int RecordMode;				// Desired record mode; see defines

	M3DAudioMotionParameter LastParameters;		// Position and orientation where
recording starts

	double RoomSize[3];			// Roomsize (x,y,z)

	DWORD AbsTime;				// Absolute time since in scene at start of
recording
};

// this is the description of the function used for PLUGINDEFINEDRECORD !!
typedef BOOL WINAPI (*M3DAudioInsertPosition)(M3DAudioMotionParameter* Position);

//Byte alignment in Borland reset
#pragma pack()

#ifdef __cplusplus
extern "C"
{
#endif

//
// These are the interface functions!!
//
// At startup:
	BOOL WINAPI EXPORT M3DAudioRequestInfo(M3DAudioPluginInfo* Info);

// Before start recording
	BOOL WINAPI EXPORT M3DAudioPrepareRecord(M3DAudioPrepareRecordInfo* Info);

// Recording
```

```
	BOOL WINAPI EXPORT M3DAudioRealtimeRecord(M3DAudioMotionRecord* Position);
	BOOL WINAPI EXPORT M3DAudioStepByStepRecord(M3DAudioMotionRecord*
Position);
	BOOL WINAPI EXPORT M3DAudioPluginDefinedRecord(M3DAudioInsertPosition
InsertFunction);

// After recording
	BOOL WINAPI EXPORT M3DAudioFinishRecord(void);

#ifdef __cplusplus
}
#endif
#endif
```

M3DAudioRequestInfo

The function

BOOL WINAPI EXPORT M3DAudioRequestInfo(M3DAudioPluginInfo* Info);

identifies the DLL and describes the possibilties which the DLL provides.

Function must return **true**, if no error occurred.

A possibile implementation could be:

```
BOOL WINAPI EXPORT M3DAudioRequestInfo(M3DAudioPluginInfo* Info)
{
	strcpy(Info->Name,   "My input plugin");		// insert Plugin Name
	strcpy(Info->Version,	"V 1.0");				// insert Version
Number
	strcpy(Info->Copyright,	"(c) 1997 My Company");	// insert Copyright
information
	strcpy(Info->Comment,	"My Plugin provides a possibility\r\n"
				"to record a motion path by simple speaking");	// insert
Program information

	Info->Capabilities =  RELATIMERECORD |
				STEPBYSTEPRECORD |
					PLUGINDEFINEDRECORD;	// insert Capabilities
(RELATIMERECORD, ...)
							// This DLL can handle all modes.

	Info->PluginIdentifier =	MOTIONPLUGIN;		// insert MOTIONPLUGIN for
a valid Motion DLL!

	return TRUE;
}
```

struct M3DAudioPluginInfo

```
struct M3DAudioPluginInfo
{
        char Name[256];             // insert Plugin Name
        char Version[256];          // insert Version Number
        char Copyright[256];        // insert Copyright information
        char Comment[1024];                 // insert short description information

        int Capabilities;           // insert Capabilities (RELATIMERECORD, ...)
        int PluginIdentifier;       // identifies the Plugin, use MOTIONPLUGIN for valid DLL!
};
```

This structure is passed through the M3DAudioRequestInfo function. It contains the elements:

Name	with the official name of the plugin.
Version	version description of plugin
Copyright	information about your copyright
Comment	a brief description of your plugin
Capabilites	defines the capabilities of your DLL. Use an OR-operation to indicate supporting multiple modes. Supported values are 1. realtime record (RELATIMERECORD) 2. step by step record (STEPBYSTEPRECORD) 3. plugin defined record (PLUGINDEFINEDRECORD) You only have to implement the corresponding functions of the capabilities. If you only want to provide a realtimerecord, set the value of Capabilities to RELATIMERECORD and write the realtime record function. You can combine the values. To provide all features use: Capabilities = (RELATIMERECORD | STEPBYSTEPRECORD | PLUGINDEFINEDRECORD);
PluginIdentifier	must always be MOTIONPLUGIN for a valid plugin.

Motion Record Plug-in SDK

To create your own DLL,or Dynamic Link Library, one needs to implement several functions and export them. The function prototypes and several declarations of required structures are located in the include file "3dplugin.h".

The required functions are:

// At startup:
BOOL WINAPI EXPORT M3DAudioRequestInfo(M3DAudioPluginInfo* Info);

// Before start recording
BOOL WINAPI EXPORT M3DAudioPrepareRecord(M3DAudioPrepareRecordInfo* Info);

// Recording; a valid DLL must implement at least one of these functions depending on its recording capabilites
BOOL WINAPI EXPORT M3DAudioRealtimeRecord(M3DAudioMotionRecord* Position);
BOOL WINAPI EXPORT M3DAudioStepByStepRecord(M3DAudioMotionRecord* Position);
BOOL WINAPI EXPORT M3DAudioPluginDefinedRecord(M3DAudioInsertPosition InsertFunction);

// After recording
BOOL WINAPI EXPORT M3DAudioFinishRecord(void);

All structures must be compiled as BYTE alignment! The header file contains two statements which will change the aligment to byte in many compilers. If your compiler does not know what it all means, search the compiler's help file for instructions on how to change the following expression:

// Set alignment to byte in Borland / Watcom:
#pragma pack(1)

//Set alignment to default in Borland / Watcom
#pragma pack()

If you build a valid DLL, please contact us, so that we could offer the DLL to our other users. Perhaps you can distribute your DLL through us!

Simple Motion-Capturing .DLL

The provided demonstration plugin allows motion capture either in realtime or by step-recording.

After setting the desired options in the "**Record Path**" dialog, hit the 'Record' button. A window will appear, showing the room and the current position of the source or head in three different views, corresponding to the XY-, XZ- and YZ-plane. You can focus any of these views by clicking on it with the mouse. Use then the joystick to change the position, or drag the source with the mouse.

Other elements in the dialog include the number of recorded steps, the amount of time remaining until the next step [key-frame], and the X-Y-Z coordinates. In addition you may specify the joystick you wish to use, and that the object traveling along this path should always maintain its relative orientation to the virtual head.

PLEASE NOTE that the item "Next time" is only available in step-by-step mode. Furthermore, this demo plug-in only supports recording translations [something which is NOT a limitation of the SDK]. Other motion classes will need to be inserted and/or modified 'by hand'.

M3DAudioPrepareRecord

The function:

BOOL WINAPI EXPORT M3DAudioPrepareRecord(M3DAudioPrepareRecordInfo* Info);

is invoked by 3D-Audio to initiate recording. A structure of type **M3DAudioPrepareRecordInfo** is passed that contains information on the desired recording mode and the initial source or head position. You can perform initialisations of your code in this function, like creating a window.

Function must return **true**, if no error occurred. Else the recording will be canceled.

struct M3DAudioPrepareRecordInfo

```
struct M3DAudioPrepareRecordInfo
{
	HWND ParentWindow;				// Handle to parent window; use this to open your window
	int RecordMode;				// Desired record mode; see defines

	M3DAudioMotionParameter LastParameters;	// Position and orientation where recording starts

	double RoomSize[3];				// Roomsize (x,y,z)

	DWORD AbsTime;				// Absolute time since in scene at start of recording
};
```

The structure contains these elements:

ParentWindow	handle to parent window. Use this handle to hook your window.
RecordMode	the desired record mode.
LastParameters	A struct of type M3DAudioMotionParameter, containing the position and orientation of the source of head at start of the recording process.
RoomSize	size of the room (x,y,z).
AbsTime	the point of time within the entire project at which the recording starts.

M3DAudioFinishRecord

The function:

BOOL WINAPI EXPORT M3DAudioFinishRecord(void);

is invoked after completion of recording. Use this function to clean up internal data and close your window.

Function must return **true** if no error occurs. Otherwise the recording will be cancelled, though it would be done by now anyway.

M3DAudioRealtimeRecord

The function

BOOL WINAPI EXPORT M3DAudioRealtimeRecord(M3DAudioMotionRecord* Position);

is called in user-defined intervals by 3D-Audio to obtain the next motion instruction. The instruction is passed in a struct of type **M3DAudioMotionRecord**. Only absolute motion seem to make sense in this mode.

Function must return **true** as long as the recording process is not finished, i.e. **false** signal to stop recording.

struct M3DAudioMotionRecord

```
struct M3DAudioMotionRecord
{
        M3DAudioMotionParameter Parameter;      // Motion instruction
        DWORD RecordTime;                       // in msec since realtime recording started

        BOOL Changed;                           // TRUE if position changed; else instruction will
be discarded
        BOOL Paused;                            // TRUE if recording is paused when realtime
mode is running
};
```

The struct contains data defining a new motion instruction to insert.

Parameter	the motion instruction itself in a sub-struct of type M3DAudioMotionParameter.
RecordTime	the time since recording started in msec. Only in realtime-mode!
Changed	set this to false, if this instruction is to be discarded because it contains no position change
Paused	set this to true to halt the flow of time given in RecordTime when recording in realtime mode

struct M3DAudioMotionParameter

```
struct M3DAudioMotionParameter
{
        double MotionParameters1[3];        // First parameter vector
        double MotionParameters2[3];        // Second parameter vector

        BOOL    FixedToHead;          // TRUE if the source is always directed to the head
        double AngularVelocity;               // For turning motion

        int       Motion;                     // Type of motion, see defines above

        DWORD TimeToEvent;                    // Time that this motion takes in msec.
};
```

The structure holds data of a motion instruction itself.

MotionParameters1	First parameter vector of motion instruction. For an absolute translation this is the target position.
MotionParameters2	Second parameter vector of motion instruction. For a relative translation this is the acceleration.
FixedToHead	**true**, if the source is to be directed towards the head during the motion.
AngularVelocity	angular velocity when using turning motion.
Motion	the type of motion. See the defines in "3dplugin.h" for available types. The expected paramter vectors correspond to the parameters available for the motion-types within the motion-dialog when manually creating a motion path.
TimeToEvent	the time that this instruction is to last. Ignored when using realtime-recording.

M3DAudioStepByStepRecord

The function:

BOOL WINAPI EXPORT M3DAudioStepByStepRecord(M3DAudioMotionRecord* Position);

is called by 3D-Audio to obtain the next motion instruction. The instruction is returned within the structure **M3DAudioMotionRecord.**

Function must return **true** as long as the recording process is not finished, i.e. **false** signals the end of recording.

M3DAudioPluginDefinedRecord

The function

BOOL WINAPI EXPORT M3DAudioPluginDefinedRecord(M3DAudioInsertPosition InsertFunction);

Is called once to render process-control to the plug-in. It passes a variable of type **M3DAudioInsertPosition**, which is a pointer to function. The plug-in can call this function to insert a new motion instruction. The plug-in shall return from this function when recording is done. To record a new instruction, call:

InsertFunction(argument);

where "argument" is of type M3DAudioMotionParameter*.

Function must return **true**, if no error occurred. Otherwise, the recording will be cancelled.

Conflicting Pinnae Cues

Pinnae are the cartilaginous projections of the outer ear, and are as unique as fingerprints. The brain intuitively associates the frequency effects of these projections as spatial data, because pinnae lie between the inner ear and the outside environment. These associations are further ingrained by the orientation and shape of the nose and shoulders.

3D Audio renders the effects of a set of generic pinnae upon moving sound sources, and this is the precise reason why one should listen through headphones for the full effect. In so doing, the listener 'bypasses' his own pinnae in favor of 3D Audio's. Therefore, if using loudspeakers, the listener will process the spatial information twice - once through his pinnae and once through those of the software.

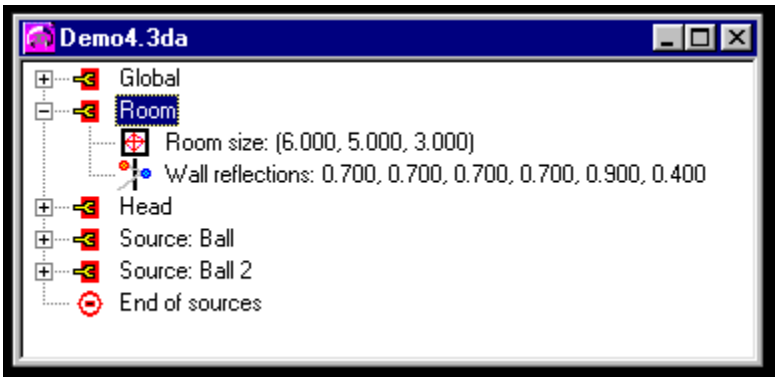
Acoustic Coloration

As with any device, a physical room imparts is own acoustic signature to the sound source. When spectrally compared with the 'dry' signal, it is precisely evident how they differ. It is this difference that acousticians determine and balance when they design a listening or recording space.

Because the brain interprets spatial data from indirect ['room'] reflections as well as the direct waves, 3D Audio must also model the room to complete its Auralisation process. If the calculations of the program are not fed directly to the inner ears via headphones, the signal will be spectrally modified by the listening room (thereby distorting the image).

File Menu

This is the menu where one manipulates project and sound files, as well as Wizard preferences.



New Creates a new document.

Open.. Opens a sound file or project file. If the **Open as read-only** box is checked, the file cannot be over-written with the **Save** function.

Close Closes the document displayed as the active MDI-child window.

Preferences

File / Preferences / Use New Wizard Enables the **New Wizard**.

File / Preferences / Use Source Wizard Enables the **Source Wizard**.

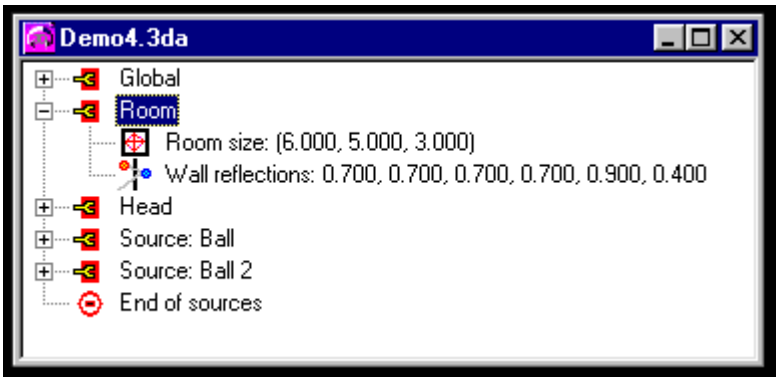
Save Saves the current document.

Save As... Saves the active document with a new filename.

Exit Closes **3D Audio**.

Table of Reflections

Here lists the number of reflected sound beams introduced by allowing for reflections of higher order. These beams must be computed in addition to the direct, line-of-sight beam:



Path-Loss Exponent

The path-loss exponent is used for computing the attenuation of the arriving sound, depending on the distance between source and head. The attenuation will be proportional to the distance to the power of the given exponent.

The natural value is -2.0, but it may lead to a too strong attenuation especially when using few reflections. In such a case you might want to lower the value to about -1.5.

This latter effect is used mainly to give a source its initial position; obviously, however, it's also a tool for special effects...

What's new in version 1.2?

Have a look at the revision history for newly-implemented features.

3D Audio also now implements a powerful new tool: motion capture!! A demo .DLL for capturing by means of mouse, joystick and keyboard is included, along with a Software Development Kit [SDK] for writing custom motion-capture routines.

Also, registered users get FREE UTILITIES!

System Requirements

The program requires Windows 95 or Windows NT 4.0 and a properly-installed soundcard!

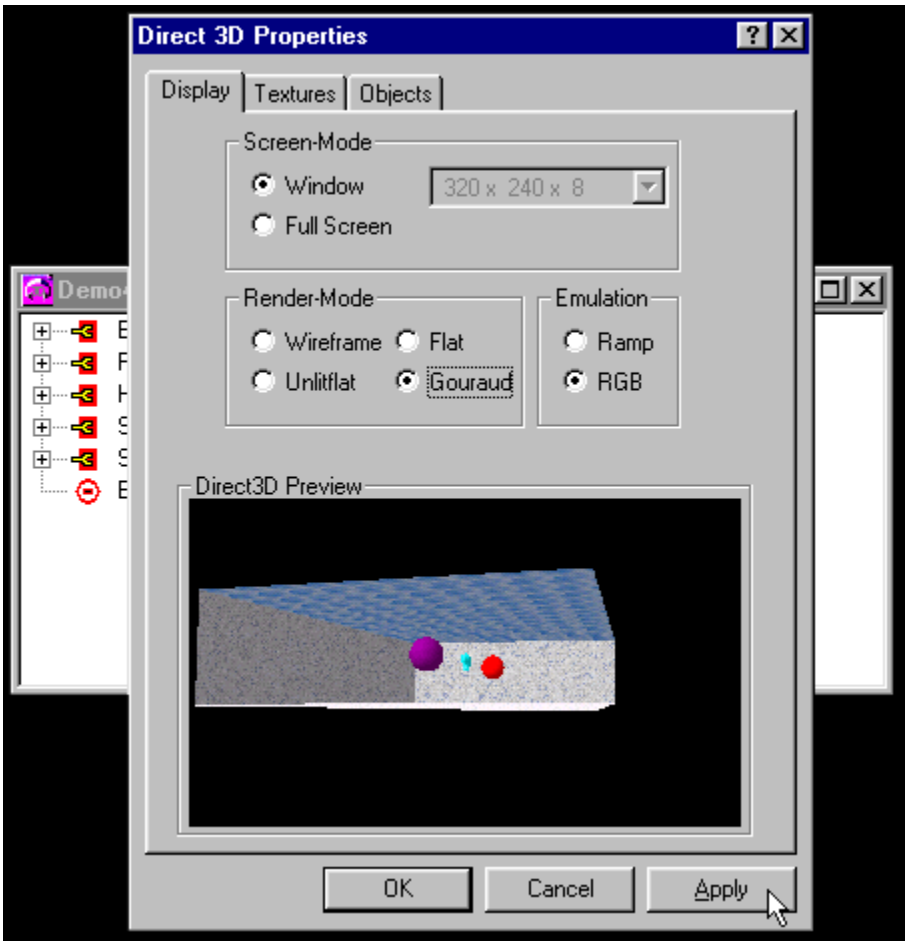
DirectX 3 is required for best-quality previewing. You may obtain a free copy of the installer from Microsoft, via this Website:

http://www.microsoft.com/msdownload/directx/dxf/enduser5.0/default.htm

Warning! Please note that this file **does not** work with Windows NT 4.0; rather, install Service Pack 3 (which includes the DirectX 3 drivers). Again, you may download from Microsoft:

ftp://ftp.microsoft.com/bussys/winnt/winnt-public/fixes/

Warning! If you use the program with Windows NT 4.0, please also read this NT 4.0 Technical Note!


Direct 3D Properties
Display
Textures
Objects
Screen-Mode
Window
320 x 240 x 8
Full Screen
Render-Mode
Wireframe
Flat
Unlitflat
Gouraud
Emulation
Ramp
RGB
Direct3D Preview
OK
Cancel
Apply

Program Status

This program is shareware, and as such may be used for seven days of evaluation. If this period is exceeded, you must register the program in order to legally use it! If you wish to distribute the program, please examine the distribution status.

In addition, one may download another demonstration project (pingpong.zip) for 3D Audio. This package demonstrates a better spatial effect, because of the larger source file as compared to the included demos in the Tutorial. You may obtain a copy from our Website:

http://www.audiophile.com/climax

Also, please have a look at the planned features page! Your suggestions are welcomed on this next-generation product.

Motion Capture

A rather simple demonstration plug-in is included with the program; it captures absolute translations using the mouse, keyboard or a joystick. A Software Development Kit [SDK] is also provided for the development of custom **3D Audio** motion capture utilities!!