Audio Compositor

Introductory Topics

Samples, Layers, and Patches

The Realtime Engine

Rendering MIDI Files

Introduction: What Audio Compositor Is

Audio Compositor is software for realizing standard MIDI files directly as digital audio. Scores that are not too demanding can be processed in real time and played on your computer's sound card. Alternatively, the output may be directed to a .WAV file for later playback--and in this mode Audio Compositor can deal comfortably with extremely complex scores and extraordinarily large collections of input samples.

Audio Compositor can also be driven in real time from your computer's MIDI IN port--that is, it can make your computer act as a sound module that you can play from your MIDI keyboard. The program's performance in this role will vary depending on processor speed and available physical memory; on less than a fast Pentium system it does not make for a really playable musical instrument, but can still be a useful aid to patch editing.

Audio Compositor's sounds are derived from instrument samples in individual .WAV files. It does not come with a ready-made sample library--there is only a small classical guitar patch to serve as an introduction to the program. Beyond that, you must do your own collecting, or design your own sounds with the assistance of other software. (This in turn says a lot about the program's intended audience, which will not include very casual users.)

Why Use Audio Compositor?

Music production on general-purpose computers is nothing new (it predates the commercial synthesizer), and a variety of software for the purpose can be found today--perhaps the best known is Csound. The spirit of most such software is to offer synthesis techniques that aren't, and perhaps never will be, implemented in the latest popular MIDI keyboard.

Audio Compositor does not quite belong with this distinguished company--it is simply the software equivalent of that most mundane of electronic music devices: the sample-playback machine. The value of software like this has been debated in various forums, the negative argument being that software wavetable synthesis uses a two thousand dollar computer to do the job of a two hundred dollar sound card, only more slowly.

It's true, as a little experience with its realtime functions will show you, that at today's processor speeds Audio Compositor is not a very capable instrument for live performance. Still, when sending its output to a .WAV file, it can do several things that a ordinary MIDI device cannot.

For one thing, Audio Compositor can deal comfortably with sample collections occupying hundreds of megabytes, using your hard disk as virtual memory and automatically shuttling samples between disk and RAM as needed. (Indeed it does this even when played from the keyboard--that is why you notice a slight delay the first time you play a given note.) Sample memory layouts that would be very expensive in hardware thus become quite practical. Of course, like any sample-based instrument, Audio Compositor is happiest with lots of physical memory as well, but the memory it uses is your computer's regular RAM--not memory installed on your sound card where it can be used for only one purpose.

Audio Compositor also works well with extremely dense or complex scores. When its output is a WAV file, it can produce practically an unlimited number of simultaneous voices. There is no serial interface at which MIDI data may be bottlenecked, and the timing of events in the output file is accurate to the nearest sample frame. Most of the small MIDI studio's usual sources of noise and distortion are eliminated--particularly if you have the means to move the output .WAV files to DAT or recordable CD for playback.

Having said all this, there is no use pretending that Audio Compositor is a good general-purpose substitute for a wavetable soundcard, especially if your goal is to hear General MIDI files at what might be termed the standard level of quality. It is designed as a tool for musicians who're trying to stretch the limits of MIDI and traditional sampling technology, by removing memory and polyphony barriers in exchange for speed of processing.

Midi Implementation

Audio Compositor's internal architecture is quite simple. Samples (individual .WAV files) are mapped to the MIDI keyboard to form basic voices called *layers*. A *patch* in turn may contain any number of layers, which may sound simultaneously or be separated by velocity-based crossfades. Audio Compositor provides a graphical environment for designing the keyboard mappings, amplitude envelopes, controller routings, and so on.

Audio Compositor's MIDI implementation is not especially sophisticated at this point, but includes the following:

Velocity, pitch bend, pan, volume, the sustain pedal, and tempo indications are recognized in the usual way.

Velocity, channel aftertouch, MIDI note number, and all MIDI controllers are routable to amplitude, pitch, and parameters of the amplitude envelope on a per-layer basis (certain combinations may not be practical).

Program changes are recognized for 128 patches, with no bank select. To use more than 128 patches you would need to process a score in multiple passes.

Your Sample Library

Audio Compositor's raw material is your sample library, a collection of samples in Windows WAV format that you must somehow acquire. There is really no special physical "library"--the samples may be scattered around your hard disk or organized nicely into subdirectories (the latter approach is recommended!) Each sample that you intend to use in Audio Compositor must first be entered into a master list in the layer editor. If a sample must be looped, or if it will ever need to be pitch-shifted (this probably includes most of your samples), then you will need to add loop and pitch information to the WAV file using Audio Compositor's waveform editor, or any audio editor that understands this type of additional data.

Samples may have been recorded at any sampling rate (though it's generally best if they match the rate you will select for your output file). They can be mono or stereo; stereo separation will be maintained if you select stereo WAV output. The sample word size must be 16 bits--Audio Compositor does not know how to load 8 bit samples.

There are many sources of instrument samples these days. You can buy them on CD-ROMs designed for various samplers, though extracting these sounds to WAV files may tax your ingenuity. If your CD-ROM drive will read raw audio, you may find it more convenient (and more economical) to purchase samples on audio CDs and extract the contents using one of the audio grabbers available on the network. Less exotically, you can of course connect your audio CD player to your sound card and re-digitize: if your sound card is in the Turtle Beach class or better you may be surprised at how little quality you sacrifice compared to a direct digital transfer. As of this writing, CD-ROMs with samples already in .WAV format are beginning to appear on the market.

Lots of mediocre samples, and a few good ones, can be downloaded from various places around the net.

I would like nothing better than to distribute Audio Compositor together with a full set of instrument samples, but the samples I use myself are from commercial sources and are not mine to give away. What's needed is a set that is (1) complete enough to be musically useful, (2) of very high quality, and (3) in the public domain. It's easy to satisfy any two of these conditions; please contact me if you can suggest how to manage all three.

Direct To Digital

Realizing a MIDI sequence directly as digital audio has certain advantages: for example, no electronic sources of noise or distortion are present, and there is no "noise floor" whatever. But before rejoicing over the absence of cables, mixer, and effects boxes, consider what you will have to do to replace their functionality. Mixing in the basic sense must be accomplished in the context of the MIDI file itself, using velocity, volume controller, etc. Effects can only be supplied after the fact, by loading Audio Compositor's output into your favorite audio editor and applying whatever capabilities it may have. (Reverb will probably be at the top of your list.)

It would not be true to say that Audio Compositor is absolutely free of noise and distortion, since your samples will generally undergo both amplitude and pitch transformations before they are transferred to the output file. The pitch shift is the more problematic part, since doing this cleanly is a complex operation. Different algorithms are available so that you can trade off quality against speed of execution; see Resampling Algorithms.

System Requirements

Audio Compositor is a demanding application, and it is designed primarily for what not so long ago were considered high-end systems. It runs on Intel or compatible platforms and requires Windows 95, or Windows NT beginning with version 3.51. Testing on NT has not been thorough.

(An earlier release of Audio Compositor, version 1.2, will run on Windows 3.11 with the Win32s extensions, though its realtime functions are disabled when running on Win32s. Copies of this version are still available as of this writing.)

Audio Compositor requires a floating-point unit--or at any rate its performance without one will be ridiculously poor--so a 486DX must be considered the minimum processor. The program is math-intensive and responds well to faster CPUs. Realtime output is awkward below about a Pentium 90, though success here is also heavily influenced by your sound card and its driver.

The memory requirement is harder to generalize about. Ideally, of course, you would want enough physical memory for all of the samples Audio Compositor will be using at a given time. On the other hand, if you had this much memory lying about, you might be better advised to install it in your sampler, and not use Audio Compositor at all. In other words, while paging memory to disk is normally to be avoided, it is precisely this capability that distinguishes your computer from your sampler, and Audio Compositor is designed to capitalize on it. Running it on eight megs of RAM may be painful, but trading speed and convenience for sample capacity is what Audio Compositor is all about. Sixteen is much more practical, and above 32 the realtime functions become usable with large sample sets.

A sound card is a practical necessity, though the program will run without it. Wavetable capabilities are not necessary.

Packing List and Installation

Audio Compositor consists of a single executable called **Ac.exe**. The distribution also includes starter layer and patch files, and a small set of classical guitar samples, whose chief purpose is to give you something to look at when you first start up the program. There is also a small MIDI file with which you can do a test run.

There is nothing here to warrant a fancy installation program--just unzip the distribution, keeping the directory structure intact, and you will have something like:

c:\Ac\Ac.exe
c:\Ac\Ac.lay
c:\Ac\Ac.pat
c:\Ac\Leyenda.mid
c:\Ac\Samples\Guitar\Guitar p 2G
c:\Ac\Samples\Guitar\Guitar p 3C
(more for a total of 12 guitar samples . . .)

There is no reason not to use a different drive letter or directory name, but if you want the guitar patch to work you must keep the guitar samples in the same path relative to the program itself. That is, if Audio Compositor is installed in

h:\Dodgy-looking-software\Ac

then it will look for the guitar samples initially in

h:\Dodgy-looking-software\Ac\Samples\Guitar

If you find other samples in the kit, they'll also be installed in subdirectories under \Samples. When you add samples of your own, however, they may be installed anywhere on your computer (even on a different disk from Audio Compositor itself).

About the Example Files

Audio Compositor comes with a set of classical guitar samples, and the initial patch and layer files are preconfigured with a guitar patch that uses these samples. There is also a MIDI file containing an abbreviated version of Isaac Albeniz's *Leyenda*, which is designed to work with the guitar patch.

Including these example files was a tough decision. Since Audio Compositor is distributed over the network, the guitar sounds had to be compact, and they were not recorded with very good equipment in the first place. The MIDI file is quite simple. All of this is completely contrary to the main purpose of Audio Compositor, which is to allow you to put a very large array of high quality samples to use.

On the other hand, the guitar examples let you see something besides an empty window when you bring up Audio Compositor for the first time, and a few moments spent looking at them can probably tell you more about the program than an hour of reading this manual. Running *Leyenda* can also give you a quick idea of how well Audio Compositor will perform on your system. The guitar samples are also well suited to realtime control from your MIDI keyboard, since they fit into about 300 KB of memory and load quickly from disk.

The **Readme** file contains instructions for running the *Leyenda* example straight out of the box.

Registering Audio Compositor

Audio Compositor is shareware; if you like it please register it! Before registering, however, you should familiarize yourself with the program and understand how well it will run on your computer. For details see the Registration Form.

Registration activates certain features that are unavailable in the shareware version. In this version of Audio Compositor the differences are as follows:

Feature	Unregistered	Registered
Maximum sample rate (WAV output)	22050	48000
Stereo WAV output	No	Yes
Maximum realtime polyphony	6	32

Note that the increased realtime polyphony will not benefit all systems; see Realtime MIDI Performance. Audio Compositor's "offline" polyphony (when rendering a MIDI file to WAV output) is unlimited even when unregistered. Keyboard-driven realtime output is always monophonic (this restriction will be lifted in a future version).

Audio Compositor Registration Form

When you register your copy of Audio Compositor, you will receive a personal registration code which adds functionality to the program. The registration code is also valid in updates and future versions of Audio Compositor, the latest of which may be downloaded from the Audio Compositor web site:

```
http://www.eden.com/~mitchell
```

Audio Compositor 1.3 was released in July 1996; the registration terms given here are good through May 1997. After that date, please check the web site for the latest news.

Registration costs $40 US (US check or money order, or international money order drawn in US dollars), payable to Scott Mitchell. Mail with this form to:

```
Scott Mitchell
6350 N. New Braunfels Ave.
Suite 104
San Antonio, TX 78209
USA
```

Registration codes are delivered by postal mail. Please allow a reasonable amount of time for delivery, but if you have questions by all means get in touch via e-mail (mitchell@eden.com).

Thank you for registering!

```
        Name: _____________________________

     Address: _____________________________

              _____________________________

              _____________________________

     Country: _____________________________

      E-mail: _____________________________ (if available)
```

Optional:

```
    Your O/S:  ( ) Win95     ( ) WinNT

    CPU Type: _____________  Memory: _____

  Sound Card: _____________________________

  Where did you find Audio Compositor?
```

Overview of the Program

If you drop down Audio Compositor's **File** menu you'll see that it can open four types of files. The first three types are files that you can edit in various ways:

> **Sample** files, which are standard Windows .WAV files containing individual instrument samples. You will need to add loop and pitch information to most samples using Audio Compositor's wave editor (or a compatible editor). Some samples, such as unpitched percussion sounds, may need no preparation at all if they are already in 16-bit .WAV format.
>
> The one and only **layer** file (AC.LAY), which has two functions. First, it keeps a master list of the samples available on your hard disk(s). Second, it maps groups of samples to notes on the MIDI keyboard to form intermediate objects called *layers*. For example, if you have a set of six fortissimo trumpet samples at various pitches, you might create a layer called **Trumpet ff** in which the six samples are mapped across the trumpet's range.
>
> **Patch** files, with a .PAT extension. Each patch file can describe a set of up to 128 numbered patches. A patch corresponds to a MIDI program change, and is composed of any number of layers. Patches also contain most of the interesting synthesizer-like parameters that control musical expression. For example, if you have three trumpet layers called **Trumpet p**, **Trumpet mf**, and **Trumpet ff**, you might add them to a patch called **Trumpet** and specify velocity crossfades between them. Each layer could also have its own amplitude envelope, controller routings, and so on.

The fourth option is to open a standard **MIDI** file. Audio Compositor doesn't edit MIDI files; rather, you open a MIDI file when you are ready to render it. The rendering process creates a digital audio realization of the MIDI file using the samples, layers, and patches created in the previous steps. Files that are not too complex can also be played through your sound card as they are rendered.

Another component of Audio Compositor is the Realtime Engine, represented by the bar at the bottom of the main window. When the realtime engine is started, Audio Compositor becomes a MIDI module that you can play live from an external keyboard (or in a pinch, by clicking a special on-screen keyboard with your mouse). Its sound is derived from whichever wave, layer, or patch is currently active on the screen. The number of simultaneous voices you can play depends on the speed of your computer and your sound card; it's usually necessary to spend some time adjusting the settings to get clean output in this mode.

Preferences

The Preferences dialog (**Edit | Preferences**) is used to set some global parameters for Audio Compositor:

Audio Output Device

If you have more than one sound card in your system, you can choose from this list which one Audio Compositor will use. Another option is "Auto-select," which allows the system to find a device that can handle whatever sample rate is called for at the moment. If you only have one audio card, you don't need to worry about this setting.

Always Force Integer Loops

Audio Compositor's wave editor has the ability to create loops with fractional lengths. While this feature can be extremely useful at times, it can also get in your way--for example if you are editing a file for use with other software that requires whole-number loop lengths. You may also be bothered by an admittedly confusing inconsistency in Audio Compositor itself: that fractional loops are respected by the realtime engine and the MIDI-to-WAV renderer, but not by the wave editor when you press the **Play With Loop** button.

The wave editor allows you force integer-length loops for the current editing session by checking an item on its **View** menu. To make this behavior the default for all editing sessions, check **Always Force Integer Loops**.

Minimum Loop on Playback

This affects playback in the wave editor only. When you press the **Play With Loop** button, Audio Compositor has a choice between looping the sound itself, and asking your sound card to perform the loop. In many cases your sound card can do a better job of this, but using "driver" looping exclusively would defeat the wave editor's ability to let you hear the loop change as you drag the wave on screen. The compromise is to let the soundcard loop, but interrupt it at intervals with a new loop buffer. This setting defines the interval.

If you find that loops don't play smoothly in the WAV editor, try increasing this setting. (On the other hand, a few soundcards may not be able to do their own looping; if playing a looped sound hangs the program, try a setting of zero.) Some cards are happy with a value of 300; others require settings up to 6000 or above. A lower setting makes the WAV editor more responsive when you are dragging loop points.

This parameter has no effect on realtime playback.

The Wave Editor

The Wave Editor is used to prepare individual samples for use by Audio Compositor. While it resembles a general-purpose audio editor in some ways, it is specialized for looping sounds and defining their pitches--two operations which are particularly important to Audio Compositor. The loop and pitch information is imbedded in the .WAV file when you save it.

Audio Compositor's Wave Editor is intended for editing individual instrument samples, and doesn't perform well if asked to load a .WAV file that's larger than a few megs.

Playing the Wave File

 **Play, Play With Loop, Stop**

These transport-control buttons are used to play the sample through your sound card. Normally the entire file will be played, but if you have set start and/or stop markers (described shortly) then these will be observed. Also, if you have selected a region in the upper view (by dragging the mouse in it) then you will hear only that region when you press Play.

If a loop has been set, **Play With Loop** causes the sound to play continuously until you hit the **Stop** button. You can edit the loop points while this is going on. If no loop points have been set, then **Play With Loop** acts just like the regular **Play** button.

Important: When you press **Play With Loop**, the loop is performed by your sound card's driver. If you have defined a fractional loop length, it will be rounded off to the nearest sample. To hear an accurate rendition of a fractional loop you must use the Realtime Engine.

Navigating the Wave File

When you open a .WAV file in the Object Editor, you will see the contents displayed in two windows, one above the other. The lower window always displays the entire length of the file; the upper window is the main editing view and can zoom in on as small a region as you like.

By dragging the mouse in the lower, full-length view, you select the region of the file that will be displayed in the upper view. This is the quickest way to close in on a particular portion of the file, and takes the place of the traditional scroll bar.

Dragging the mouse in the upper view will also highlight a region, but this has a different purpose than selecting in the lower view--it is usually done to define a region as the target of some editing operation. However, you can drag in the upper view with the *right* mouse button to slide its contents from side to side. This is a convenient way to make small adjustments to the view.

Stereo sounds are displayed differently than in most audio editors. Instead of placing the left and right channels in seperate windows, Audio Compositor displays them together on the same axis, in different colors. The left channel is always shown in black. (Sometimes, the right channel may not even be visible except at higher magnifications.)

Rescaling the View

 **Horizontal Zoom, Unzoom**

The magnifying-glass buttons increase and decrease the horizontal scale of the upper window, generally by a factor of two. An exception is that if you select a portion of the wave in the upper view and then hit the "magnify" button, the selected portion will expand to exactly fill the upper view.

It's possible to zoom in so closely that spaces opens up on the screen between individual samples. When this happens, gray tick-marks appear along the bottom edge of the view to show you where individual samples are. The image of the wave is then drawn in a connect-the-dots fashion, which is mathematically dubious but convenient when you are doing precise looping.

 **Vertical Zoom, Unzoom**

These buttons increase and decrease the vertical scale, allowing you to view waves with small amplitudes. The scale changes by a factor of two with each click.

Setting Start and Stop Markers

 **Set Start, Set Stop**

When Audio Compositor uses a sample in its output, it normally assumes that it can use the entire length of the file. You can change this behavior by setting start and/or stop markers. Audio Compositor will ignore material that precedes the start marker or follows the stop marker. The Wave Editor will also respect the markers when you listen to the sample.

To set a start or stop marker, click the **Set Start** or **Set Stop** button (the mouse pointer changes to remind you what you're doing) and then click in the *upper* wave view at the appropriate spot. When the markers are set, Audio Compositor puts a gray background behind the portion of the wave that won't be played. The editor will annoy you with error messages if you attempt to do something irrational like placing the stop before the start, or placing either marker inside the sample's loop.

It doesn't make sense to have a lot of material outside the start and stop boundaries--it will be occupying memory at run time even though Audio Compositor will never use it. However, there may be times when you are uncertain about how much of an instrument's attack or decay you wish to include. Moving the start and stop markers gives you a chance to experiment without actually deleting material from the sample. When you have settled on the ideal start and stop points for a given sample, you can use **Truncate to Start/Stop Markers** (on the **Edit** menu) to discard the surplus material.

Setting Loop Points

 **Set Loop Start, Set Loop End**

To introduce a loop into a new sample you must use these buttons, which work just like the **Set Start** and **Set Stop** buttons (see the previous page). The loop start and loop end points will be shown as green and red lines, respectively, in both the upper and lower views.

Use these buttons as a crude way of placing the loop approximately where you want it. You will then want to refine the loop using the Loop View described next.

Switching to Loop View

 **Loop View**

Once loop points have been set, this button toggles between the normal display and a special **Loop View**. In Loop View mode, the upper window is divided in half by a vertical line. The loop start point is drawn first at the center line, and the display proceeds from there to the right-hand edge of the screen. Then it wraps around from the left in such a way that the loop end point falls again at the center line. The effect is to show you what the loop looks like as you roll across the critical junction between the loop points.

To change the loop points, simply drag the wave from side to side with your mouse. Dragging the right half of the screen changes the loop start, and dragging the left half changes the loop end. You can watch the loop markers in the lower view to see how the loop is moving with respect to the whole file.

It's important to note that the horizontal **Zoom** and **Unzoom** buttons still work in Loop View. If you need to make very small adjustments to the loop points, you will want to zoom in closely so that you can slide them back and forth in fine increments. Conversely, you can back off for a wider view when you want to make large adjustments quickly.

When making short loops, it frequently happens that you find the ideal length for a loop before you find its ideal placement. In this case you would like to be able to change the loop start and stop points in tandem, keeping the loop length constant while trying out different positions. Dragging in the upper view with the *right* mouse button provides just this feature by locking the left and right panes together.

See also:

About Fractional Loops
More Notes On Looping

About Fractional Loops

When you create a loop with Audio Compositor, the start point is always an integer, but the end point can include a fraction--that is, it can fall between two samples. This can be useful for tuning short, single-cycle loops, or for avoiding stubborn artifacts in longer ones. Creating a fractional loop is simple: put the display in Loop View, zoom in until there is space between individual samples (hash marks appear along the bottom of the window), and move the loop end point as usual by dragging the left-hand pane with the left mouse button. The end point will stay where you put it, even if it falls between two of the hash marks.

In contrast, you will notice that when you drag the right-hand pane to change the start point, the display always jumps exactly from one sample to the next. That is because the loop start point must always be an integer.

Important: To hear the effect of a fractional loop in Audio Compositor's wave editor, you must use realtime playback (see About the Realtime Engine). The wave editor's **Play With Loop** button sends output directly to your sound card, rounding the loop length off to the nearest sample.

Fractional loops are always respected when you play Audio Compositor from an external keyboard, and when rendering a MIDI file.

To prevent fractional looping for a particular sample, pull down the wave editor's **View** menu and check **Force integer loops**. If you would like this behavior to be the default for all wave editing sessions, check **Always force integer loops** in the Preferences dialog.

More Notes On Looping

Whatever you may have learned elsewhere about looping instrument samples will probably translate pretty well to Audio Compositor. If you can find very long samples, the importance of the whole subject may be diminished, since Audio Compositor may not have to resort to looping as often as a regular sampler with limited memory.

Audio Compositor's MIDI-to-WAV rendering process has another technique for reduced looping. A normal MIDI instrument doesn't know when you are going to take your finger off of a key, so once it begins looping a sound, it must continue to do so until the key is released. Audio Compositor, however, looks ahead in your MIDI file, and knows in advance how long each note is held. As a result, it can stop looping whenever it sees that there is enough material in the sample to finish the note without further looping. (Actually it is not quite omniscient and allows a safety margin against the possibility of an upward pitch bend, but that is a fine point.)

While the less-frequent looping is a good thing in general, it can lead to an occasional unpleasant surprise if you are accustomed to thinking that material following a loop end point will never be heard.

Setting the Pitch

Audio Compositor will need to know the pitch of most samples. The pitch is specified as a musical note (C4 is middle C) plus or minus a fine-tuning amount expressed as so many cents. The cents value can range from -50 to +50 (a value of +51 would be expressed as 49 cents down from the next higher half-step). The pitch is displayed in the status bar at the bottom of the editing window, both in the musical notation just described and as a frequency in Hz.

It's not necessary to set a pitch value for unpitched percussion instruments or sound effects that are meant to be heard always at their original pitch. (You'll set the "Fixed Pitch" flag for these samples in the Layer Editor.)

Setting the pitch does not modify the sample's audio data in any way. You are telling Audio Compositor what the pitch *is*, not what it should be. In other words, if the sample is nominally an A4 but was played 5 cents flat, the pitch should be set to A4 *minus* 5 cents so that Audio Compositor can compensate when the sample is performed. Changing the pitch setting will have no effect on the sound you hear when the sample is played directly by the Wave Editor.

The **Edit** menu's Guess Pitch function is the simplest way to set a sample's pitch initially. If further adjustments are needed they can be made using the **Pitch Dialog**, which is called up by the button shown here:  (that's supposed to be a tuning fork). The pitch dialog has three sliders which enable you to specify the sample's pitch in terms of octave number, note, and cents as described above.

The pitch dialog will remain visible until you close it, or until the current sample loses the input focus.

Compatibility with Other Audio Editors

Audio Compositor uses standard Windows .WAV files. Samples that have neither loop points nor definite pitch (most drums, for example) can be created using nearly any audio editor. Audio Compositor expects 16-bit samples at any sampling rate.

When you add pitch and/or loop points to a sample, they are stored in the .WAV file along with the audio data. The format of this information is again standard; programs from other manufacturers that are specifically designed for editing instrument samples will be compatible with Audio Compositor, though some of them may round fractional loop lengths to integer values. (If this frequently presents a problem, check **Always force integer loops** in the **Preferences** dialog to make Audio Compositor's wave editor consistent with your other software.)

Unfortunately the sampler data format (specifically I mean the "smpl" RIFF chunk) is not recognized by most general-purpose audio editors. Even the "looping" capability frequently seen is usually designed for playlist-style use and is saved in a format that's entirely different from that of sampler-oriented loops. Since these otherwise admirable products often discard information that they don't understand, you may find your loop and pitch information missing after using one of them to modify a sample. (Audio Compositor, by the way, makes a reasonable effort *not* to discard unfamiliar data placed by other editors.) Since software features are constantly changing, the best way to sort all this out is to experiment with the particular editors you have on hand.

The Edit Menu

The Wave Editor's **Edit** menu provides the following functions:

Gain Adjust
Normalize
Normalize L/R independently

Clear start/stop points
Clear loop points
Clear pitch
Truncate to start/stop points

Swap channels
Remix stereo
Reduce to mono

Crossfade loop
Resample
Change playback rate
Synthesize
Guess Pitch

Gain Adjust

Calls up the Gain Adjust dialog to increase or decrease the amplitude of all or part of the sample. This function will also do fade-ins and fade-outs.

Adjust Gain

This function adjusts the amplitude of the currently selected region, or of the entire sample if no region is selected. The adjustment is specified in decibels.

If you enter different values for the starting and ending adjustments, then the amplitude is changed by a value that changes smoothly across the affected region.

For stereo samples, you can specify different values for the left and right channels. The sliders are ganged for convenience--if you need to set them to different values you must work from right to left.

Normalize

Increases the amplitude of the entire sample by the greatest amount possible without clipping.

Normalize L/R independently

Increases the amplitude of both channels of a stereo sample, each by the greatest amount possible without clipping.

Clear start/stop markers

Removes the start and stop markers from the sample if they are present. (More accurately, it sets the start marker to zero, and the stop marker to the end of the sample.)

Clear loop points

Removes loop points from the sample if they are present.

Clear pitch

Removes the sample's designated pitch. An unpitched sample always sounds at its actual pitch, rather than being transposed according to the note that triggers it.

Truncate to start/stop points

Deletes any material in the sample file which precedes the start marker or follows the stop marker.

Swap channels

Quickly reverses the left and right channels of a stereo sample.

Remix stereo

Calls up the Remix stereo dialog, which allows you to adjust the relative levels of the left and right channels, or to reduce the stereo seperation between channels.

Remix stereo

This function can be used to rebalance stereo samples or to reduce stereo separation. It replaces each channel with some mixture of the two original channels.

The two sliders on the left determine how the new left channel will be created. Setting this pair to 0 and -96 dB, respectively, will leave the left channel unchanged; raising the second value (and reducing the first as necessary) will introduce some right-channel content into the left channel. Applying this operation symmetrically to both channels at once (e.g., setting the sliders to -2, -12, -12, -2) has the effect of narrowing the stereo image of the sample.

This is probably not an operation you will want to perform very often, but it's proved useful in rare cases when a particular stereo sample sounds unusually "wide" and stands out from its neighbors.

Reduce to mono

Calls up the Reduce to mono dialog, which allows you to mix a stereo sample down to mono.

Reduce to mono

This function changes a stereo sample to mono, mixing the two channels together in the proportions you specify. To save only the left channel, set the sliders to 0 and -96 db; for an equal mixture of left and right, set both sliders to -6 dB (or a bit higher if the two channels contain dissimilar material.)

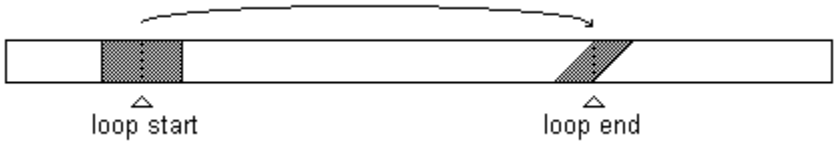
Crossfade loop

Calls up the Crossfade loop dialog, which allows you to perform a destructive crossfade around the current loop points.

Crossfade Loop

Audio Compositor does not perform crossfade looping on the fly. This function performs a "hardcoded" crossfade, permanently modifying your audio data. Obviously this is not an ideal solution, but as some samples simply cannot be looped without crossfading, this option is provided as a last resort.

The portion of the sample that is modified is centered around the loop end point:



The region surrounding the loop start point is used as source material. It is mixed into the region surrounding the loop end point, beginning at a low level, rising to 100 per cent at the loop end point, and then subsiding again to zero. The effect is that of a traditional crossfade loop when the sample is heard in loop mode. When the sample is played straight through, there are really two crossfades in quick succession as the transplanted material emerges and then recedes. Depending on the nature of the sound, this effect can be anywhere from imperceptible to totally unnatural.

Two parameters must be specified:

Radius

> The duration (in thousandths of a second) of the fade-in leading up to the loop end point. The total length of the effect is twice this value (or possibly less, since the second fade may be accelerated if there is not enough material after loop end to accomodate it at full length).

Contour

> Crossfading a loop often creates a glitch in apparent sound volume around the point of the crossfade. If the fade is done linearly, then a relatively long crossfade on a complex sound like a string section can exhibit a dip in amplitude. An equal-power fade will correct this, but conversely may create a "bulge" when applied to other types of material. Set "Contour" to zero for a linear fade, to 100 for an equal-power fade, or to some value in between.

Since the Wave Editor has no "undo" function, be sure the original sample is saved to disk before experimenting with crossfade looping. Try a variety of settings to see which values work best, using the **File** menu's **Reopen** function each time to discard the changes. Be sure to listen to the result both in loop and straight-through mode. Short fades (a few milliseconds) are often useful for loops with stubborn clicks, while longer fades can help when looping unstable section or chorus sounds. When you have decided on the best settings, save your work.

It should be clear that if you create a crossfade in a sample, and later change the location of its loop points, the crossfade will no longer work properly. A good practice is to keep a backup copy of the original, unadulterated sample. Nondestructive crossfading is on the Audio Compositor "wish list."

Resample

Calls up the Resample dialog, which allows you to resample or pitch-shift the sample.

Resample

This function changes the sample rate of your audio data, and optionally changes the file's nominal playback rate to match the new sample rate. If the playback rate is not changed, the operation is referred to here as "pitch-shifting" and you must specify the new pitch. The result of this operation will only be correct if you have set the pitch of the original sample accurately. That is, if the current pitch is G4 and you request a shift to A4, the sound will be transposed up one step whatever its real frequency.

If the playback rate is changed along with the audio data, then the apparent pitch of the sound will remain the same and we use the term "resampling." In this case you simply specify the new rate.

The available resampling methods are described under Resampling Algorithms.

You should be aware that some sound cards cannot play sounds at non-standard sampling rates. 44100, 22050, and 11025 Hz are of course the safest bets. There are also cards (including at least one prominent and expensive model) that play non-standard rates but at severely degraded quality.

Note that Audio Compositor never requires you to resample manually, since it can accept input material at any sample rate.

Change playback rate

Calls up the Change playback rate dialog, which changes the nominal playback rate of a sample without modifying its audio data.

Change Playback Rate

This function changes the file's nominal sample rate without modifying the audio data.

Synthesize

Calls up the Synthesize dialog, which can produce a few simple waveforms.

Synthesize

This dialog allows you to generate some simple waveforms. Calling this "synthesis" is going a bit far--it's included here mainly as a curiosity, though if you are patient and know exactly what you're doing you can manage a crude sort of additive synthesis with it. (One word: detune.) Expanding this feature for serious use really falls outside the scope of this version of Audio Compositor, but I'm open to suggestions.

Replace existing sample

If you check this box, the contents of the current WAV file will be discarded and replaced by the new material.

Add to existing sample

The new material will be mixed with your existing sample.

Length

The length of the new sample, specified either in milliseconds or in cycles. If you choose the latter option, the length will be so many periods of the frequency set in the **Pitch** box. The **Length** options are grayed out if you have checked **Add to existing sample**, because in this case the current length is never changed.

Pitch

This specifies the frequency of the new waveform, either as a musical pitch (e.g. "C 4" or "A#5") or in Hz.

Waveform

Pretty slim pickings.

Envelope

This determines the amplitude of the signal as a function of time. The envelope is drawn with the mouse--the procedure is identical to drawing an Amplitude Envelope in the Patch Editor.

Guess Pitch

Despite its tentative sound, the **Guess Pitch** function often works very well. It estimates the pitch of a small section of the sample. Normally this section is in the very center of the file, but if you select a portion of the wave with your mouse, Audio Compositor will focus its attention near the center of your selection.

Guess Pitch produces a message like the following:

> The estimated pitch is D#3 -15.29 cents. Change to this value?

Answering "Yes" sets the pitch of the sample to the proposed value. Alternatively, you can tell **Guess Pitch** that it guessed too high or too low; in this case it will try to produce another estimate. Because **Guess Pitch** tends to be either wildly off the mark or quite accurate, the process usually takes only a few iterations, and you will not need an especially good ear to tell you whether a reasonable guess has been arrived at. Of course, **Guess Pitch** is not infallible, and certain types of sounds confuse it completely. In these cases your ears become more important!

Since **Guess Pitch** operates only on a small section of the file, it may not work well for samples whose pitch is unstable. For example, samples with vibrato can give varying results depending on where the selection is placed. Taking several estimates from different parts of the file may produce a useful average.

The Layer Editor

The Layer Editor does two related jobs. In the "Samples" area of the window, it keeps a list of the samples that are available on your computer. Each sample is assigned a "friendly" name, and the sample list displays this name rather than the full directory path to each sample's .WAV file.

The "Layers" area is used to create, copy, and delete layers. Clicking on the keyboard at the top of the screen will map the currently selected sample to the currently selected layer. Double-click to unmap a key.

Mapped keys can be "played" by clicking with the right mouse button. This requires some patience as a key will not speak immediately the first time it is pressed. A better way to audition a layer is to use the Realtime Engine.

The Layer Editor stores all of its data in a single file called **Ac.lay**, in Audio Compositor's home directory.

Maintaining the Sample List

The sample list is maintained using the **Add**, **Edit**, and **Delete** buttons, which hopefully are self-explanatory. Double-clicking the name of a sample is a shortcut for the **Edit** button. A fourth button toggles the sample list in and out of a special tree view.

Adding or editing an entry in the sample list involves the following settings:

Path

The *full path* to the .WAV file containing this sample. You can type it in by hand, or use the **Browse** button to locate the file. If you move this file at a later time, you will need to edit this field accordingly, or Audio Compositor will not be able to find the sample.

Name

Each sample is given a short name in addition to its full file specification. This makes the layer editor a little easier on the eye. The recommended practice is to name the sample after its file--that is, if full path of the WAV file is

e:\Samples\Trumpets\Trumpet F 5 mf.wav

then the sample should probably be named "Trumpet F 5 mf". For new entries, Audio Compositor will practice this convention for you if you leave the default "Untitled" and then use **Browse** to locate the WAV file.

Level

This setting gives you a way to balance the sample's volume against others that will be mapped to the same layer; it is given in dB and is normally set to zero.

Fixed Pitch

Check this box if you don't want the sample transposed to match the particular note that triggers it. This is appropriate for unpitched percussion instruments, sound effects, etc.

Ignore Key-up

Check this box if you want the full length of the sample to be played out regardless of the length of the note that triggers it. This behavior can be useful with certain percussion sounds. You can get the same effect by setting a very long release phase in the amplitude envelope, but setting **Ignore Key-up** is slightly more efficient.

This setting is applied only during the realization of a MIDI file--it has no effect when you are listening in the editor. It has no effect at all on looped samples (since they would play forever it it did).

See also:

The Sample List Tree View

The Sample List Tree View

If your collection of sounds grows quite large, the sample list can become inconvenient to use, since you may have to scroll through hundreds of entries to find a particular sample. In a case like this you would like to be able to organize your samples into groups, with a group for each instrument or type of instrument.

The sample list's "tree view" mode is designed for this purpose. Pressing the **Tree View** button once activates this mode; pressing it again restores the usual flat alphabetical list. The tree view organizes your samples according to their positions on disk. It looks a bit like the left-hand side of the Windows Explorer, with everything but Audio Compositor's .WAV files missing. If you have established an orderly system of directories and subdirectories for keeping your samples, it will be reflected in the tree view. Conversely, if your samples are spread all over with no rhyme or reason, the tree view will look thoroughly incomprehensible!

Path names beginning with a dot are treated as a special case; the tree view shows them as belonging to the "Audio Compositor Home Folder," which represents the directory where Ac.exe is installed. The guitar samples that come with Audio Compositor are designated this way to make the installation process simpler, but you do not need to use this convention for your own samples.

Maintaining the Layer List

When you select a layer in the "Layers" box, its keyboard mapping is displayed across the top of the window. A couple of additional parameters are displayed inside the box. To modify the keyboard map, see Mapping Samples to Layers on the next page.

When you have modified a layer, you must press the **Save** button if you want to keep the changes. Alternatively you can press **Save As** to save the edited layer under a new name. Saving a layer does *not* write the changes to your hard disk--when you are finished editing and saving one or more layers, you must then save the entire layer file using the regular Windows **Save** command from the **File** menu. This two-level arrangement may sound hazardous, but Audio Compositor will prompt you if it thinks you're forgetting a step.

The layer list has no "Add" button. The way to create a new layer is to start with an existing one--usually the "Empty Layer," which is always present--modify it, and then save it under a new name using the **Save As** button.

To summarize the buttons in the Layers box:

Save

Commits any changes you have made to the currently selected layer.

Reset

Rolls back any changes you have made to the currently selected layer since you last pressed **Save**.

Save As

Saves the current layer, but under a new name. The original layer remains unmodified.

Delete

Deletes the current layer (but not its component samples).

Clear

Unmaps any samples from the keyboard, making a clean slate of the currently selected layer.

In addition to its keyboard map, a layer has the following properties:

Level

Use this to adjust the amplitude of the layer relative to other layers. It is given in dB and is normally set to zero.

Comment

A plain text field that you can use in any way you like.

Mapping Samples to a Layer

Clicking on the keyboard maps the currently selected sample to the currently selected layer. A sample can of course be mapped to any number of individual keys. Double-click on a key to remove the mapping.

Audio Compositor regards middle C as C4.

You'll see that as you add samples to the keyboard, it changes color. Gray keys have no sample mapped to them; the other colors have no individual significance, but they alternate to show you the split points between different samples. If you move the mouse over a mapped key, the name of its sample is displayed just below the keyboard.

Auditioning a Layer

The best way to audition a layer is to start realtime input and play it from your MIDI keyboard. Apart from being more "musical," this method plays looped samples cleanly, even with fractional loop lengths. The first time you play a note in each zone of the keyboard map, you will notice a slight delay as the corresponding sample is loaded from disk.

If you don't have a MIDI controller attached to your computer, try the Virtual MIDI Keyboard.

If realtime output simply doesn't work on your computer, you can audition the layer by clicking the layer editor's own keyboard with the *right* mouse button. Again, there may be a slight delay, and if you release the button before the sample is loaded, you won't hear anything at all until you right-click again on that key.

Samples played using the layer editor's on-screen keyboard accumulate in memory: the amount of memory tied up in this way is displayed in the "Cache" box. If you overdo it and your system becomes paralyzed, hit the **Clear Cache** button to purge some of the samples from memory. For a more thorough housecleaning, press **Clear All**.

If realtime playback is running then memory management is handled by the realtime engine, and controls in the "Cache" box have no effect.

Right-clicking on a key that has no sample mapped to it (a gray key) does nothing.

Layer Editor Menu Options

The Layer Editor has just a couple of its own menu options:

Edit | Reparent file pointers
Reports | All samples
Reports | Samples for current layer

Reparent File Pointers

If you have collected a large number of samples, you will have invested a great deal of time in telling Audio Compositor where to find each one. If you later decide to move some or all of your .WAV files to a different directory, or to a new disk drive, you could be faced with the rather mind-numbing task of editing the file path specification for hundreds of samples.

This function is intended to ease such situations. Please note that it does *not* cause your .WAV files to be moved from one place to another. It's used when you have moved the .WAV files by hand, and need to correct the now-invalid pointers in the sample list.

Reparent file pointers is very simple and rather dangerous: it simply scans the entire sample list for .WAV file paths that begin with the "Old Root," substituting the "New Root" in its place. A typical case might be:

> Old Root: c:\Ac\Samples
> New Root: d:\Samples

This is entirely a string-replacement chore, and Audio Compositor will not try very hard to stop you from entering completely irrational values. Be sure to check the results thoroughly before saving your work.

Reports | All Samples

This produces a report describing all of the samples known to the layer file. It gives the name of each sample and the full path to its .WAV file, and in addition it opens each .WAV file and reports information about its pitch, loop points, etc. For large sample collections this can be a fairly slow process.

If you have an entry for a sample whose .WAV file has been deleted or moved, this will be noted in the report. Thus, running a report is a convenient way to validity-check your sample list.

The report is called Samples.txt and is created in the current directory. For convenience, Audio Compositor will try to open it for you when it's ready using Write.exe (which on Win95 usually means WordPad).

Reports | Samples for current layer

This report is identical to the All Samples report, but includes only samples referenced by the currently selected layer.

The Patch Editor

Patches are at the top of Audio Compositor's "architecture"--they are the objects that correspond to MIDI program changes. Use the Patch Editor to create a patch and assign layers to it (the layers must already have been created in the Layer Editor). The Patch Editor also assigns amplitude envelopes, controller routings, and the like.

A set of up to 128 patches is stored in a "patch file," whose filename has a .PAT extension. When you make a production run with Audio Compositor, you will tell it which patch file to use. If your MIDI files are geared for several different setups then you might want a patch file to match each of them--otherwise one patch file might be all you ever need.

The Patch Editor also allows you to play sounds directly from your MIDI keyboard. This is a new an relatively experimental feature.

Details on using the patch editor:

The Patch Editor Tree View

The patch editor is divided vertically into two panes. The left-hand side gives you a "tree view" of the patch file's contents. A plus sign next to an item indicates that it has "children," and clicking on the plus sign brings the children into view. Expanding a patch item reveals its component layers; expanding a layer reveals the samples that are mapped to it.

The right-hand pane shows details about the currently selected patch and layer. It is described in more detail on subsequent pages.

The Patch Editor Toolbar

The Patch Editor's toolbar has four buttons. Note that these functions are duplicated on the **Edit** menu.

 **Create Patch**

Click this button to create a new patch. The Patch Editor creates an empty patch called "New Patch" at the nearest convenient patch number. To change the name and number to the values you really wanted, edit them in the right-hand pane of the editor window, and press the Apply button in the top-right corner.

 **Add Layer**

Add a layer to an existing patch. The layer is chosen from a list of layers that have been defined in the layer editor.

You can control the order of layers within a patch by remembering that new layers are always added at a point immediately following the selected item in the tree. If you select a patch and press **Add Layer**, the new layer will become the first layer in that patch. If you select an existing layer, the new layer will be inserted just below it.

 **Delete**

If the currently selected item is a patch, delete the patch. If a layer, unmap that layer from its "parent" patch. You can't delete a sample using this view (use the Layer Editor to unmap a sample from a layer).

 **Change Layer**

Place a different layer in the position occupied by the currently selected one. (The new layer will assume all the per-layer parameters that were applied to the previous one.)

Patch Parameters

There are only a few parameters that apply to a patch as a whole, and these are visible at the top of the Patch Editor's right side. Note that this row of items has its own **Apply** button. A patch (not a layer or sample) must be selected in the tree view before these fields can be edited.

Number

The program change number for the patch, which must be between 0 and 127. You may change this number, so long as the new number is not already in use.

Name

The name of the patch, entirely for your own information.

Level

This is used to modify the patch's amplitude relative to other patches. It is given in dB and is normally zero.

Changes made to these items do not become effective until you click the **Apply** button.

The Audition feature is described on the next page.

Auditioning a Patch

The best way to try out a patch is to start the realtime playback engine, and play the patch using an external MIDI keyboard. If you haven't got an external controller handy, try the Virtual MIDI Keyboard. As a last resort, you can click in the patch editor's own keyboard, though this method is somewhat inefficient and can only play a few seconds of sound at a time.

Whichever method you use, all of the per-layer parameters are factored in when you audition a patch, so you can hear the effects of your amplitude envelopes, crossfades, etc. Note however that changes to the layer parameters won't be effective until you have pressed the **Apply** button below the Layer Parameters tab sheet.

Per-Layer Parameters

The Per-Layer Parameters occupy a tabbed sheet in the lower-right part of the Patch Editor. This is where most of the interesting synth-like parameters in Audio Compositor are set. You can edit these only when a layer (not a patch) is selected in the tree view.

When you work with this part of the program it is tempting to think that you are editing a layer. Bear in mind, however, that the settings here apply only to an instance of the layer: if you include the same layer in more than one patch, you can assign it different parameters for each patch. You can even assign the same layer twice to the same patch, with two sets of parameters--if you can think of a reason to do so.

The parameters are divided into four groups:

- Amplitude Envelope
- Key Routings
- Crossfade
- Controller Routings

See also:

- Routing Examples
- Crossfade Examples

Amplitude Envelope

The amplitude envelope is specified by a series of coordinates, each with an amplitude expressed in dB and a time value expressed in thousandths of a second. On the screen, the envelope is displayed as a set of line segments connecting points which you can move with the mouse. Each of these movable points is marked with a small circle. The rules are:

> Any point can be moved by clicking and dragging, though the first point is "glued" to the left-hand edge of the window.
>
> You can create a new point by clicking anywhere except over an existing point. The envelope can have any number of segments.
>
> Delete a point by double-clicking on it.
>
> The last (rightmost) point is regarded as a "hold" value; it determines the amplitude of the note between the time the last envelope segment plays out and the time the note is released. It's displayed as a horizontal dashed line extending to the right-hand edge of the display.
>
> The envelope's release stage is *not* shown. Its length (again in milliseconds) is simply typed into the "Release Length" box.

With the magnifying-glass buttons, the horizontal scale can be changed to accommodate envelopes of various lengths. A problem arises when the view is scaled to show longer periods of time: the resolution of the graph becomes finer than the resolution of your monitor, so that it may become impossible to position the mouse on exactly the value you want. For example, you may want to set a segment at exactly 9000 ms, but the mouse skips from 8978 to 9007. You probably shouldn't be worried about such minute details, but if it really matters, try dragging with the *right* mouse button. This will cause the point to move by single units, lagging behind the mouse cursor if necessary.

Other values set on this page:

Level

> The relative level of this instance of the layer; the default value is zero.

Release Length

> The length of the single release segment, which is never drawn on the graph.

First Segment Linear

> Generally this should be checked. A long-winded explanation follows:
>
> On your screen, the envelope's y-axis is marked in decibel units, uniformly spaced, and the straight lines representing the segments are accurate within this context. That is, what looks like a straight line on the graph will be an exponential curve if you examine the output with an audio editor. In general, this type of response works well because it sounds like a smooth progression to the human ear, but the attack stage of the envelope is more often than not an exception.
>
> Personally I suppose this is because the attacks of real-world instruments do not often resemble exponential shapes; if you've seen visual representations of many instrument

sounds perhaps you'll agree. At any rate, a casual scoping of three popular MIDI keyboards has convinced me that synthesizer designers also treat the attack stage of the amplitude envelope as a special case, and give it a linear shape (or nearly so) rather than an exponential one. Checking **First Segment Linear** causes Audio Compositor to behave this way.

The conclusions expressed here are a bit tentative and (as with any other aspect of Audio Compositor) comments are welcome.

Key Routings

A *routing* is a linkage between incoming MIDI data and some aspect of a note's performance; the implementation here is similar to that in most MIDI instruments. Audio Compositor distinguishes between "key" routings and "continuous" routings. Key routings are evaluated once at the beginning of each note, and may affect such things as its volume, pitch, and envelope, but once the note has begun to sound they have no further effect. Continuous routings are those that can alter the note while it is sounding. To keep the distinction clear, the two kinds of routings are assigned separate pages in the parameters dialog, but they are very similar in most respects.

Routings are one area in which Audio Compositor somewhat outshines the average synthesizer. The usual method is to dial in a set of parameters that say, for example, "Please vary the attack stage of the amplitude envelope from two seconds to zero as velocity increases." The attack length will thus decrease in a straight line as you go from minimum to maximum velocity, so at a velocity of 64 the attack has no choice but to be one second long. In contrast, Audio Compositor allows you to draw the routing graphically, so that you are no longer confined to straight-line relationships.

To add a key routing, click the **Add** button, and select a source and destination. If the source is a MIDI controller, you will have to specify its number; if the destination is a stage of the amplitude envelope, you will have to specify which stage. When the routing is first added, it will have a neutral shape--that is, Audio Compositor draws a "do nothing" function to get you started. To make the routing useful, modify the function by drawing with the mouse (the procedure for drawing is the same as for drawing the amplitude envelope).

The routing source is always displayed on the x-axis, usually with a range of 0 to 127 to represent the range of possible velocity or MIDI controller values. When the pitch wheel is used as a modulation source, its value is "coarsened" to a number in the same range by dropping the least significant bits. You do not need to route pitch bend expressly to pitch, as Audio Compositor performs this by default (and without dropping any bits).

The source called *Key Number* refers to the position of the note on the 127-note MIDI keyboard. When this source is used, the vertical lines on the graph are spaced at 12-unit intervals (that is, at octaves). All of these lines represent C's on the keyboard, and the one drawn in red is middle C (MIDI note number 60).

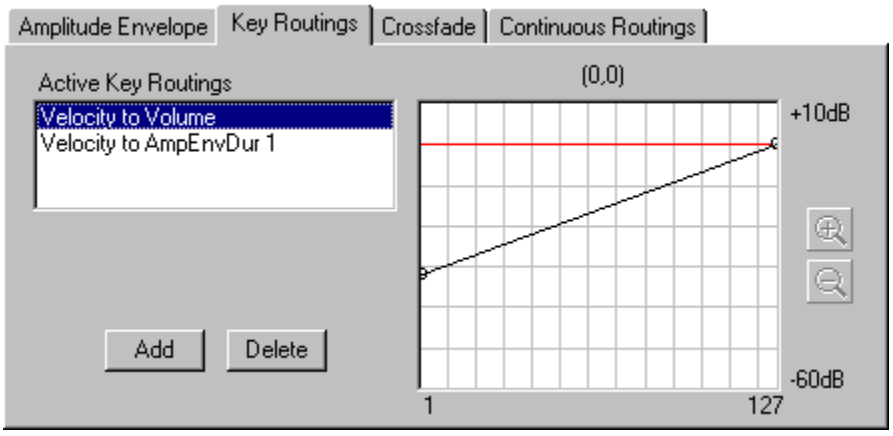
The y-axis units likewise depend on the destination of the routing. If the destination is either amplitude, or the level of a stage of the amplitude envelope, then the y-axis shows a dB value which is added to the destination. If the destination is the length of an envelope segment, the y-axis shows a *multiplier* ranging from 0 to 200 per cent.

The *Pitch* destination refers to a deviation within the current pitch-bend range, and is bounded by -100 and +100 per cent.

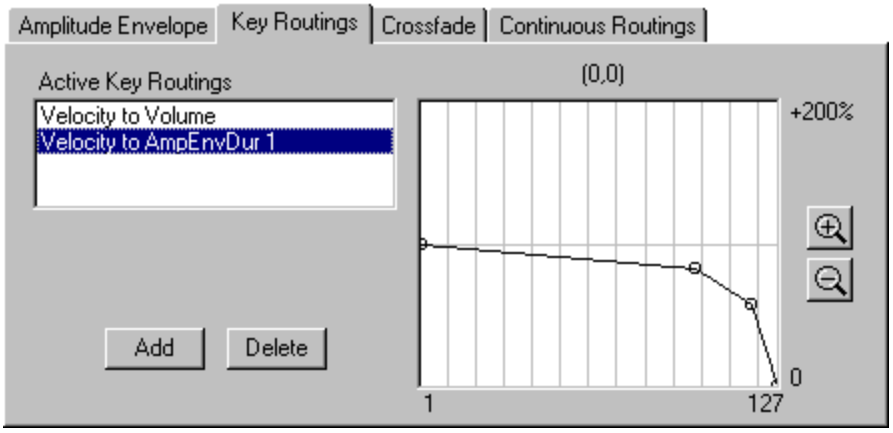
See also Routing Examples.

Routing Examples

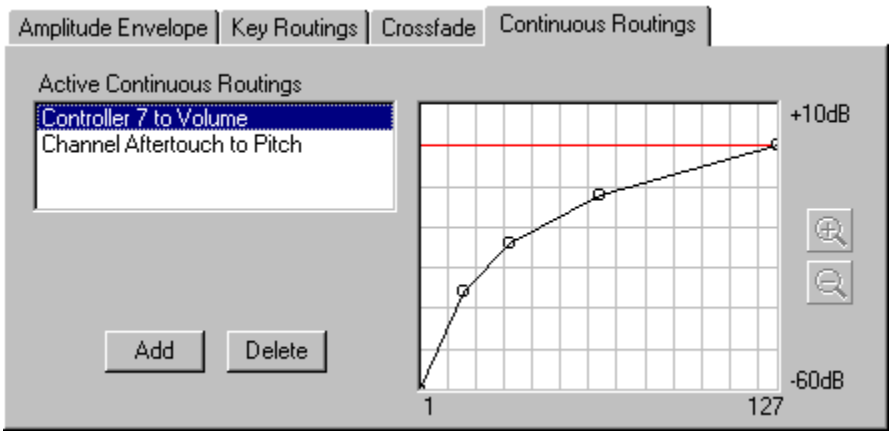
For most layers you will probably want route velocity to volume (amplitude) as in this first example. If you want Audio Compositor to mimic the "feel" of a particular MIDI keyboard, you might find that an upwardly-convex shape is more accurate than the straight line shown here. Note also that while commercial keyboards often have 60 dB or more of dynamic range, their factory presets often use less than half of that.



In the next example, velocity is used to shorten the first stage (stage 1) of the amplitude envelope. The effect is slight at lower velocities, and increases steeply at the highest velocities.



In the last example, controller number 7 is routed to volume. Since Audio Compositor will always perform this routing by default, you don't really have to add it unless you want a response that's different from the built-in function--which happens to be exactly the curve shown here. Note that the examples above were "key" routings, but this is a "continuous" routing:


Amplitude Envelope
Key Routings
Crossfade
Continuous Routings
Active Continuous Routings
Controller 7 to Volume
Channel Aftertouch to Pitch
Add
Delete
+10dB
-60dB
1
127

Crossfade

This is a velocity crossfade in the usual sense. The fade is between the current layer and the one after it (this happens to be the only way in which the order of the layers within a patch is ever important). Note that applying a crossfade to the last layer in a patch does not really make sense: the layer will fade out over the specified velocity range, but there will be no following layer on which to perform a corresponding fade-in.

The crossfade is specified by its:

Center

> The velocity at which both layers are attenuated equally. A value of zero disables the crossfade altogether, and is the default setting.

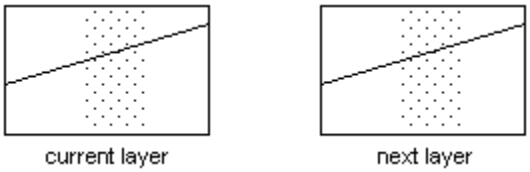
Width

> The number of velocity steps over which the crossfade takes place. Half of this range lies on either side of the center velocity. A value of zero causes an abrupt transition from one layer to the next (sometimes called a "cross-switch" in synth parlance).

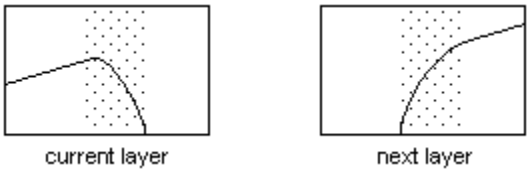
Type

> The algorithm used to calculate the crossfade. *Equal Power* uses a constant-sum-of-squares method and is the normal setting. The *Exponential* option, which is of questionable value and is not even very accurately named, yields a slightly different shape. The *Linear* option provides a straight-line fade, and may be appropriate in certain special circumstances.

In effect, a crossfade modifies the velocity-to-volume routing of the current layer and the one following it. You could redraw these routings yourself, but setting a crossfade is more convenient. If the original routings look like:



Then setting a crossfade will make them sound like this:



Note that if the current and following layers have different velocity-to-volume curves to begin with, the crossfade may not work normally.

See also Crossfade Examples.

Crossfade Examples

Crossfades may be "stacked," so if you have sampled a piano at four different dynamic levels you might have an arrangement like this:

Layer	Crossfade Center	Crossfade Width
Piano p	60	0
Piano mf	84	0
Piano f	116	0
Piano ff	0	0

If a layer with no crossfade appears in the middle of the order, the layer after it plays normally (unless it has a crossfade of its own). In the following example, the first layer crossfades into the second, but the third layer is always heard:

Layer	Crossfade Center	Crossfade Width
Violins p	80	20
Violins f	0	0
Solo Violin	0	0

Finally, this rather spectacular example shows three different solo trumpets, each sampled at three dynamic levels, combined to form a unison section patch. At low velocities we'll hear the three **mp** samples, and then with increasing velocity cross through the three **mf**s to the three **ff**s. The center velocities are staggered slightly to further disguise the transitions:

Layer	Crossfade Center	Crossfade Width
Trumpet 1 mp	72	16
Trumpet 1 mf	104	16
Trumpet 1 ff	0	0
Trumpet 2 mp	76	16
Trumpet 2 mf	108	16
Trumpet 2 ff	0	0
Trumpet 3 mp	80	16
Trumpet 3 mf	112	16
Trumpet 3 ff	0	0

If the logic here seems obscure, keep in mind that a crossfade suppresses its own layer at velocities above the fade, while suppressing the *following* layer at velocities below it. Since no crossfade is set on the **ff** layers, the effect "starts over" with each of the **mp** layers.

Continuous Routings

Continuous Routings are used in very much the same way as Key Routings; in fact if you have read about the latter there is not much to add. To repeat a distinction made earlier, key routings ignore input which is received after the beginning of a note, while continuous routings can modify the note as it plays. Consequently, the sources and destinations available for routing are somewhat different in the two cases. For example, velocity cannot be the source of a continuous routing because it is measured only once, at key-down. The length of an envelope segment can't be the destination of a continuous routing, because that segment may already have been played out at the time a control value is received.

Note that Audio Compositor routes the Pitch Wheel to Pitch by default. If you add this routing yourself (for example, you want to invert the normal action of the wheel) then the new routing will replace the default behavior. Unless you have something really bizarre in mind, you should take care to cross the x-axis exactly at a value of 64.

About the Realtime Engine

When you click the **Play** buttons in Audio Compositor's wave editor, the current wave is sent in large blocks directly to your sound card. Roughly the same thing happens when you play sounds from the layer and patch editors using their on-screen keyboards. So far, Audio Compositor behaves much like any conventional audio application.

However, Audio Compositor is also capable of producing a continuous stream of audio that reacts to MIDI events--that is, it can work more or less like normal MIDI module that you can play "live" from an external keyboard (or from a special on-screen keyboard). The component that does this job is called (for want of a better term) the *realtime engine*, and it is represented by the control bar at the bottom of Audio Compositor's main window. The **Start** and **Stop** buttons turn the realtime engine on and off. You wouldn't want it running all the time, because it puts a considerable load on your CPU.

When the realtime engine is running, any wave, layer, or patch that you're currently editing can be played from a MIDI keyboard attached to your computer. In fact, if the patch editor is the active window, Audio Compositor even responds multitimbrally to program changes on all 16 MIDI channels. If you don't have an external MIDI controller, the **Keyboard** button pops up an on-screen keyboard that you can play with the mouse.

Audio Compositor's realtime performance varies widely depending on the speed of your processor and the latency characteristics of your sound card, so the realtime engine must be adjusted to run on your particular system. The Settings button opens a window with controls for this purpose. Stuttering and distorted playback are signs that you need to experiment a little (or maybe a lot) here. The controls allow you to limit the quality of realtime output--e.g. polyphony, sample rate, and keyboard responsiveness--in exchange for more reliable operation.

Realtime output is always monophonic. Stereo output would reduce polyphony by nearly half, so it has seemed reasonable to defer it to a future version of the program. When playing stereo samples via the realtime engine, you'll hear only the left channel.

Realtime output is intended mainly as an aid to patch editing, though on some systems it makes for quite a playable musical instrument. In certain cases it is even possible to drive Audio Compositor from a sequencer running on the same computer, via a MIDI loopback device. Arrangements like this will become more practical as processor speeds increase.

Starting Realtime Output

Realtime output is controlled by the buttons at the bottom of Audio Compositor's main window. The **Start** button is grayed out unless some playable object (wave, layer, or patch) is currently open. Once you have started realtime output, you can change the focus from one window to another; the realtime engine will play whichever window is active at note-on. If you close a window, realtime playback shuts down.

The realtime engine's behavior varies a bit depending on what kind of object you are playing:

Wave Files

When the Wave Editor has the current focus, Audio Compositor maps the wave file across your entire keyboard. There is no velocity response (because that is not a property of an individual sample), but the sample's loop and pitch will be observed. If no pitch is set, the sample will be played at its actual pitch no matter which note is played. The Wave Editor responds to all 16 MIDI channels.

Layer Files

When the Layer Editor has the focus, realtime MIDI input is directed to the currently selected layer. It responds to all 16 MIDI channels.

Patch Files

When the Patch Editor is the active window, MIDI input is directed to the patch *or* layer that is currently selected in the left-hand pane. Clicking a patch or layer causes it to be selected on all 16 MIDI channels. However, program changes received via external MIDI are channel-specific. That is, the Patch Editor acts like a multi-timbral synth if you send it program changes via the MIDI IN port. (If you change to a program number that is not defined in your patch file, you will get silence!)

The first time you play a given note on your keyboard, Audio Compositor must load the corresponding sample from disk. This often introduces a noticable delay. You can "preload" the patch by touching a key in each sample's key-split zone, and then waiting a few seconds. (If the patch has velocity splits, remember that it may load different samples depending on how hard you strike the key.) Patches which require more than your computer's available physical memory will probably not be playable--if your system begins to bog down, close the active window to unload the samples.

If you have several windows open at once, be aware that the Layer and Patch editors use cached copies of individual samples. If you open and edit a .WAV file, the change may not be reflected in a layer or patch that was already open elsewhere on the screen. (The MIDI file renderer sidesteps this problem by making you close all other windows before it will open.)

Realtime MIDI Performance

Audio Compositor does not use the synthesis facilities on your sound card, only its digital-to-analog converters. All aspects of the sound production are performed on your computer's CPU--a decidedly inefficient method. Consequently, when played from an external keyboard, Audio Compositor cannot produce as many simultaneous voices as a dedicated MIDI module. (Note that this is different from MIDI file rendering, where Audio Compositor's polyphony is unlimited.)

Furthermore, while a normal synthesizer knows in advance how many seperate voices it can generate, Audio Compositor doesn't. The number depends on the speed of your system, on certain characteristics of your sound card and its driver, and on the amount of any unrelated activity on your system (even mouse movement). When you push Audio Compositor too far (by playing too many notes at once) the result is stuttering, distortion, or an interesting dive-bomber effect that may constitute a whole new synthesis method. When you first experiment with this feature, keep the volume turned down low!

Audio Compositor manages this situation in two ways. First, a slider in the Realtime MIDI Settings window allows you to set a hard limit on Audio Compositor's polyphony. Once you have learned where the boundaries are on your system, you can use this to keep Audio Compositor from overtaxing itself if you accidentally play too many notes at once.

Second, a variety of other controls in the same **Settings** window allow you to reduce Audio Compositor's workload in exchange for sound quality and musical expression. With many sound cards (especially the less expensive kind) you may need to increase the **Scan Interval**. You can also reduce the output sample rate, or tell Audio Compositor not to perform amplitude envelopes or to ignore certain types of MIDI data.

Realtime MIDI Settings

This window controls Audio Compositor's response to live input from your MIDI keyboard. Most of the parameters here are meant to help you tailor the program's performance to the capabilities of your computer. For additional information see Realtime MIDI Performance.

Select MIDI IN

This is a drop-down list of all the MIDI input devices that Audio Compositor can find on your system. You must select the appropriate one (probably an external MIDI IN port) before Audio Compositor can receive data from an external controller.

Output sample rate

The sampling rate at which audio will be sent to your sound card. At the lowest rate (11025 Hz) sound quality will naturally be poor, though on slower systems only this rate may be practical. On faster machines you may be able to sustain an adequate number of voices at higher rates. In any case, there is no point in using a rate higher than that at which your samples were recorded.

MIDI receive

Here you can selectively discard various types of MIDI data (controllers, pitch bend, channel aftertouch, and program change messages). To discard a given type of message, *clear* the corresponding check box. The effect on Audio Compositor's efficiency will depend on how the MIDI events are used in the patch you're playing.

Poly aftertouch is always ignored in this version of Audio Compositor.

Amp Envelope

This check box tells Audio Compositor whether to apply the patch's amplitude envelope (it therefore has no effect in the wave or layer editors). Turning off envelopes can improve the program's response noticeably--obviously at the expense of musical expression.

Scope

This checkbox activates the MIDI status box at the bottom of the window, which reports each MIDI message as it arrives (subject to the "MIDI receive" filters). It is a useful gadget when you are troubleshooting a problem, but turning it off will help performance.

Priority

Adjusts the scheduling priority of the realtime playback thread. A setting of +1 will make realtime output more resistant to interference from system activity such as mouse movement. Higher settings may be necessary if you want to minimize Audio Compositor while driving it from another application such as a sequencer.

Output level

Adjusts the output level within a range between 0 and -24 dB. This is a digital gain control for controlling clipping in Audio Compositor's output, *not* a volume control tied to your sound card's amplifier. If you are hearing distorted output, try reducing this

value before concluding that you are overdriving the CPU--partcularly if you are playing lots of simultaneous voices.

Polyphony

As you will soon discover, realtime MIDI polyphony is self-limiting, in the sense that the audio output goes to pieces if you push it too far. It can be useful to ask the program to limit its polyphony deliberately to some specific value, since this can save you from inadvertantly over-driving the system by playing too many notes at once. It also helps by cutting off the decay tails of notes as you play in order to stay within the limit. This voice-stealing proceeds on a first-in, first-out basis.

The maximum setting of 32 voices is probably over-optimistic as of this writing (if you find you need a higher value please write--and send a picture of your computer!). As on most synthesizers, multi-layered patches use up multiple voices in the available polyphony.

The unregistered shareware version of Audio Compositor has a maximum polyphony of 6.

Scan interval

Note: this parameter has no effect on Audio Compositor's MIDI file rendering process, which is always carried out with single-sample accuracy even when audible output is turned on.

This is the interval at which Audio Compositor scans the MIDI input for new MIDI events. It's expressed in milliseconds. Ideally you would like to set it to 1, so that Audio Compositor could respond that quickly when you play a note on the keyboard.

Unfortunately, the "scan" is also the interval at which Audio Compositor interrupts your sound card to pass it fresh output, and many sound cards fall into conniptions when beaten on at anything close to a 1-millisecond rate. Even with a speedy card, a figure of 10 is more realistic. At this setting, the delay before a played note sounds becomes noticeable, but not hopelessly objectionable. A MIDI sequencer with a resolution of 75 ppq at a tempo of 80 would degrade timing to about the same extent.

Even more unfortunately, the figures mentioned above are still out of reach for many sound cards, particularly the most affordable ones. To overcome stuttering or distorted output, you may need a setting of 30, 50, or even 80. Unless you are a cathedral organist, you will find the delay introduced by these settings very disagreeable, though realtime output can still be a useful aid to patch editing.

Default pitch bend

The default pitch-bend range, expressed in half-steps. (A value of four means two up and two down). It's currently not possible to modify this value via MIDI (write if this is important to you). It has no effect on efficiency.

Slow driver

Some soundcards and drivers respond more quickly than others to system requests. If you get stuttering or distorted playback, and increasing the Scan Interval doesn't help, try checking this box. The downside of this feature is that it will loosen up Audio Compositor's timing a bit.

Experience has shown that most garden-variety sound cards are "slow," so this box is now checked by default when Audio Compositor is first installed.

Save settings

If you check this box, Audio Compositor will remember the current settings (except for the **Save settings** box itself) the next time you start it up.

MIDI IN Channel Routing

This window allows you to route each channel of incoming MIDI data (i.e. from your keyboard) to Audio Compositor, or to a MIDI OUT port of your choice, or to both at once. This makes it possible, for example, to layer Audio Compositor with your sound card's internal synth. The default setting is to send all MIDI channels to Audio Compositor, and none to MIDI OUT.

If you select a MIDI OUT port but do not assign any channels to it, and then close the Channel Routing window, Audio Compositor will close the MIDI OUT port. This avoids tying up a port for no reason.

The Virtual MIDI Keyboard

Surely there is a special place in the next world for people who misuse the word "virtual," but it seems inescapable here. Pressing the **Keyboard** button at the bottom of Audio Compositor's main window pops up a keyboard control that you can play with your mouse--not a wholly satisfying musical experience, but useful at times if you do not have the real thing connected to your computer.

From Audio Compositor's point of view, clicking on the Virtual MIDI Keyboard is exactly equivalent to playing notes on an external MIDI controller. Both send MIDI events to the realtime engine, which you should read about before using this feature. Until you have tailored the realtime engine's performance for your computer and sound card, its output may not be acceptable.

The keyboard responds differently to the left and right mouse buttons: the left plays individual notes, while the holding down the right button plays any note that the mouse passes across. The "sustain" checkbox works like a pedal--this is particularly useful with the wave editor, since it allows you to leave a note playing while you drag loop points around. Uncheck the box to turn off any held notes. The **Velocity** slider determines the velocity that's sent with each note-on.

When you are editing samples, layers, and patches, the best way to audition them is via MIDI--using an external controller if one is present, or the "virtual" keyboard described here. Playing Audio Compositor's sounds via MIDI is more memory-efficient than other methods, and it's also the only way outside the MIDI file renderer to hear the effects of fractional loop points.

Sounds can also be played directly from the wave editor (using the control buttons) and from the layer and patch editors (using the clickable keyboards that appear in those windows). Use these methods if you simply can't get the realtime engine working properly on your system.

The MIDI File Renderer

When you open a MIDI file in Audio Compositor, your object is not to edit it but to render it as a .WAV file, or as live output to your sound card (or both). The MIDI file renderer displays your MIDI tracks as horizontal bands. Nothing happens if you click your mouse in this window, which is simply a big progress indicator.

A session with Audio Compositor's MIDI file renderer goes something like this:

> Click the **Job Setup** button. This brings up a dialog in which you define the characteristics of the rendering job, such as the name and characteristics of the output file, how much of the MIDI file you want to include, etc.
>
> Click the **Go** button, and sit back (or perhaps turn in for a good night's sleep) while Audio Compositor does its work. Depending on a variety of factors, you may also be able to listen to the program's output while it is running.

For details see the following pages:

MIDI Rendering Toolbar and Menu

 **Job Setup**

Opens the job-setup pages, which define what will happen when you press **Go**. See Setting Up a Rendering Job.

 **Go**

Starts the rendering job.

 **Stop**

Terminates a job prematurely, closing the output file.

 **Purge Memory**

Pressing this button while a job is in progress causes the sample cache to be deallocated. This is not useful ordinarily, but may come in handy if you realize too late that you don't have sufficient pagefile available for the current job. See Memory Management.

 **View Cache**

This button removes the progress grid from the main window, replacing it with a text view of the sample cache. This lists each sample currently in memory, its reference count (the number of voices using the sample at a given instant), and its size in bytes.

 **Audible Output**

Asks Audio Compositor to send a copy of its output to your sound card. The sound will be discontinuous if Audio Compositor cannot calculate it at "real speed." If Audio Compositor is running *faster* than real speed, then turning on audible output will slow it down.

Menu Items

For the most part, Audio Compositor's menus just duplicate the toolbar functions. The one exception is the **View** menu's Warnings option.

Mapping Failure Warnings

Organizing program-change messages in the MIDI file can be error-prone with Audio Compositor. Most normal synthesizers come out of the box with a patch ready to play at each of the 128 program numbers, and bad changes are often easy to spot--a cymbal crash where a flute was expected, for example. Audio Compositor is different: particularly in the hands of new users, the patch database is likely to be "sparsely populated," with most of the patch numbers vacant. Patch-change miscalculations thus result only in silence, and may be hard to diagnose.

Activating the **Warnings** option on the **View** menu can be helpful in diagnosing missing output; it is also a good way to scan for subtle errors in a large file that is substantially correct. When this option is active, the MIDI file renderer stops and issues a warning when it finds a MIDI note that doesn't map to any sound. A note can fail to map for several reasons:

> No program change was ever specified on the note's MIDI channel, and no patch is defined in the patch file for program 0 (which is the default).
>
> A program change was specified, but it was for a number that is not defined in the patch file.
>
> A valid patch was found for the current program number, but none of its layers includes a mapping for this particular note. (For example, you played a C7 in your tuba patch.)

In the warning box, the current patch may be reported as "0 or vacant". This can mean either that the patch was set (or defaulted) to zero, or that a program change was issued for some other number that was not defined in the patch file.

Note that Audio Compositor numbers patches from 0 to 127, so if you are working with a sequencer that counts from 1 to 128 you will need to compensate.

Preparing the MIDI File

Audio Compositor reads Standard MIDI Files in the usual Type 1 format. The number of tracks is currently limited to 64, but raising this number is trivial and I'll do it if anyone requests it.

There are certain types of MIDI data that Audio Compositor ignores. One of these, Polyphonic Aftertouch, represents a real gap in Audio Compositor's MIDI implementation. Sysex data is also ignored.

There is one serious defect in Audio Compositor's approach to MIDI files, but you may find it either annoying or useful, depending on your habits. While Audio Compositor is processing a given track in your MIDI file, it is oblivious to all the other tracks (except the tempo track). This means that a program change or MIDI controller on one track never affects the playback of notes on another track, even if they share the same MIDI channel. Many musicians deliberately seperate same-channel data onto more than one track for a variety of special purposes, knowing that at performance time it will all be streamed together down a single MIDI cable. If you use this technique, you will need to merge such tracks before rendering your MIDI file with Audio Compositor.

On the other hand, many musicians use multiple MIDI interfaces to overcome the 16-channel limitation of the MIDI specification. If you are placing multiple tracks on the same MIDI channel and then routing them to multiple MIDI interfaces to gain "virtual" channels, Audio Compositor plays right into this approach--every track is effectively on its own channel(s).

Note that the converse of the situation just described is not a problem: if you mix several MIDI channels on a single track, Audio Compositor will keep them properly seperated.

Setting Up a Rendering Job

After you have opened a MIDI file in Audio Compositor, there are still some questions to be answered. Do you want to process the whole file, or only a bit out of the middle? Do you want to process all the tracks at once, or just a selected few? What should the sample rate of the output file be, and do you want it in stereo?

To answer these questions, choose **Settings** from the **Edit** menu, or click the shortcut button on the toolbar:  This calls up a tabbed dialog with several sections:

Files
Tracks
Range
Constants
MIDI

Files

You may either type these values in (the full path is required) or use the **Browse** buttons.

Output File

The .WAV file that Audio Compositor will write. If this file exists, the new material will be mixed into it.

You may also leave this item blank, in which case no output file will be written. This presumes that you're planning to listen to the output in real time--otherwise, Audio Compositor will be exercising your computer to no purpose. Declining an output file can speed up real-time output considerably, if all you want to do is listen.

Patch File

The patch file to be used in this run.

Layer File

The layer file to be used in this run. This defaults to **Ac.lay** in Audio Compositor's home directory, and should only be changed by advanced users with some special purpose in mind.

Tracks

Here you will find a list of the MIDI file's tracks, and you can select any combination of them by highlighting with the mouse. If "Render Selected Tracks" is checked, then Audio Compositor will make one pass through the file, rendering only the highlighted tracks.

For relatively simple scores it may be practical to select all of the tracks, and have Audio Compositor render the entire piece in one go. There are two reasons to do differently--one is that you might just want to hear a subset of the tracks seperately from the rest of the score. The second reason is that Audio Compositor may work more efficiently on larger scores if you break the job up "horizontally" into groups of tracks.

One way to do this is to render a subset of tracks and then, without necessarily exiting from Audio Compositor, select some or all of the remaining tracks and render them to the same output file. The new tracks will simply be mixed with the old.

By sacrificing some flexibility, you can ask Audio Compositor to automate this procedure for you. Checking the **Process All** option tells Audio Compositor to process the file in blocks of continguous tracks. In this mode, if the MIDI file has twenty tracks and you select tracks 5, 12, and 20, then Audio Compositor will make three passes through the file, rendering first tracks 1 through 5, then 6 through 12, and finally 13 through 20. Since no intervention is required between passes, this gives you a way to "batch" process a large composition.

Deciding how many tracks to process at once requires some experience. More is involved than sheer numbers--you should also consider the relationships between tracks. If you have 1st violins, 2nd violins, and violas on separate tracks, and if (good viola sounds being in short supply) they all use some of the same samples, then it makes sense to render them together in order to take advantage of Audio Compositor's caching.

Range

By default, Audio Compositor will render the entire length of the MIDI file; this page lets you restrict it to a certain section. The beginning and ending points are specified in the Measure/Beat/Tick format common to most sequencing programs. Note that measures and beats are counted from 1, but the first tick is numbered 0.

The range is also shown in "wall time." If you change the M/B/T values, you can press the "Recalc" button to refresh this part of the display.

The **End of Track Padding** setting tells Audio Compositor to continue for so many milliseconds after the end point is reached. This is necessary when you are processing the whole length of a score, because the end-of-track markers in the MIDI file may not give time for the release segments of your envelopes to complete. (If you plan to run the output through some other audio editor in order to add reverb, add a few more seconds of padding to make room for the final decay.)

If you are sending the output of several seperate runs to the same file, it is up to you to ensure synchronization by setting the same beginning point each time. (Unless you exit the program, Audio Compositor will remember these settings from one run to the next, so this is not as inconvenient as it sounds.)

Constants

The values on this page govern either the characteristics of the output file, or Audio Compositor's internal behavior:

Maximum Sample Cache

This parameter limits the amount of memory that Audio Compositor will allocate for input samples. The out-of-the-box default is 8,000,000 bytes, which is adequate for small projects. Setting it lower than available physical memory will cause unnecessary page swapping. If you have sufficient pagefile space, you may increase it drastically (100 megs would not be at all unreasonable, though it would be unnecessary if the total size of your sample collection is not that large).

If you set this value higher than the amount of available virtual memory (free physical memory plus available swapfile space) then you may get out-of-memory errors while running Audio Compositor. On Windows 95, you can avoid this by keeping this setting well below the amount of free space on the hard drive where your pagefile is located. For a more complete discussion, see Memory Management.

Output Sample Rate

The sample rate of the output file. If you are adding material to an existing file, be sure to set the same rate as was used to create the original file. The maximum output rate is 48000 (or 22050 in the unregistered version of Audio Compositor).

Output Buffer Size

This sets the size of Audio Compositor's output buffers. It is measured not in bytes but in samples, and there is more than one buffer, so the actual amount of memory allocated is several times the number you enter here. 32768 is a good value. Small values (a few thousand) may improve efficiency by conserving physical memory, while larger values may improve real-time rendering if you are memory-rich. The effects will vary between computer systems.

Granularity

This parameter tells Audio Compositor how often to recalculate a sound's amplitude when it is changing over time in response to an amplitude envelope. (I believe it's similar to what Csounders call "k-rate".) It is expressed in samples--that is, at a 44100 Hz output rate, a value of 44 means that recalculation will take place just over 1000 times per second.

Ideally this parameter should be set to 1, making amplitude envelopes completely smooth. You will probably want to reserve this degree of perfectionism for final takes, since it can slow down processing severely. For values greater than one, the segments of your amplitude envelopes become step functions rather than straight lines, and the difference is heard as distortion at fast rates or "zipper effect" at slower ones. The effect is dramatic with a pure sine wave as source material. With real instrument sounds, it often becomes far less noticeable.

Attenuation

This is the master gain control for MIDI file rendering--it is combined with the "level" settings found throughout the various editors, to determine the amplitude at which

each sample will be mixed into the output file. If the level settings are zero at every stage (sample, layer, and patch), then setting **Attenuation** to zero as well would mean that samples are transferred to the output file with no change in amplitude. Recommended practice is to keep the levels of the various components centered around an average of about 0 dB, then use the **Attenuation** setting to avoid clipping in the output file. You will need to experiment to find the right value. Very simple scores may need just a few dB of attenuation; dense ones as much as 24 or more.

Note that this parameter works oppositely from the level settings--a positive attenuation value *reduces* the amplitude, so it will normally be set to a positive value.

Pitch Bend Range

This sets the pitch bend range globally. A value of 4 means that the range is two half steps on either side of center. This version of Audio Compositor does not provide pitch bend range control for individual channels.

MIDI

This crude view of the input file's raw MIDI data is included simply as a convenience (and one of questionable value at that). The format is rather cryptic, but the MIDI-literate will be able to make out most of it. The data cannot be modified. MIDI data that Audio Compositor ignores, such as Poly Aftertouch and Sysex blocks, will not be visible because Audio Compositor has already discarded it from memory.

Running the Job

After you have set up the job parameters, click "OK" to dismiss the Job Setup dialog. The portion of the MIDI file you have selected for rendering is shown by cross-hatched areas on the main window.

Click the **Go** button to start the job. The cross-hatched parts of the window turn solid as the program makes progress. When the job is finished a box pops up which gives some statistics about the amount of work that was done; the most interesting number here is probably the "ratio"--this is the ratio between the time spent processing and the duration of the output file.

When you have dismissed the little statistics box, your MIDI file is still loaded and you can immediately set up another run if you wish.

To listen to the output, you must open the output WAV file using another application such as the Windows Media Player or your favorite audio editor. This version of Audio Compositor does not know how to play back the sound from your output file after it has been processed. Audio Compositor's own WAV editor is not designed for files larger than a few megs, so it won't normally be useful for playing your finished productions.

Performance Considerations

Since Audio Compositor makes relatively heavy demands on any computer system, it is natural to ask what can be done to configure it for best performance. The stubborn fact is that in most applications, the program spends the great majority of its time doing sample-rate conversion (i.e., pitch-shifting) and fine-tuning its other operations is not particularly rewarding.

The most important performance factors, then, are the speed of your processor, and the resampling algorithm you choose on the **Constants** page of the job setup dialog. For details about the latter see Resampling Algorithms.

Memory issues are of secondary importance, but you will need to know more about them if you are working with large projects. See Memory Management.

Resampling Algorithms

Audio Compositor can use either of two different resampling algorithms to transform samples when they must be pitch-shifted, or when their own sample rates do not match that of the output file you are creating. Choosing the algorithm means trading off quality against speed of execution.

Note that these options apply only to the MIDI file renderer. Live input from a MIDI keyboard is always produced using linear interpolation.

Linear Interpolation

If you don't have a background in digital signal processing (and I don't), linear interpolation is the method you might devise on the back of a napkin if asked to change the sample rate of a signal. The original signal is treated as a series of straight lines connecting adjacent samples, and the new signal is created by sampling this fictitious shape at the new rate.

While this method has no mathematical respectability at all, it works surprisingly well for smaller ratios--a few half steps--so long as the sound has no very prominent high-frequency components. It is also fast (though this is of course a relative term). Using linear interpolation on a high-pitched glockenspiel sample is a good way to reveal its limitations: the upper harmonics may alias at all sorts of unpleasant frequencies and the fundamental may even seem to disappear altogether. But with most kinds of sounds, you will probably find the method good enough for serious work.

Band-limited interpolation

This is far slower than the linear method. A lowpass filter is applied to remove frequencies above the Nyquist limit of the new rate. It will take some experimentation to decide whether you want to use this method. You are likely to find it is only needed for particular sounds, and there may even be cases where it can make things worse (because the filter is not perfect). A nice enhancement to Audio Compositor would be the ability to assign a resampling method to an individual sample, rather than choosing it globally for an entire rendering job.

Memory Management

When Audio Compositor is running, it allocates memory on the fly each time it needs to load a new sample file. When it is finished with a particular sample, this memory is not immediately freed, since the sample is likely as not to be needed again before too long. The collection of allocated samples is referred to here as the *sample cache*, and it can grow quite large. During MIDI file rendering, the **Maximum Sample Cache** parameter (specified on the Job Setup **Constants** page) limits the cache size; when the maximum is reached Audio Compositor will try to free less-recently-used samples so as to stay within the limit. This effort may fail if all of the currently cached samples are in use--meaning that they are all sounding at the instant in your MIDI file that is currently being rendered.

When you are working interactively with Audio Compositor's various editors, there is no particular limit on cache sizes; if your system bogs down or you see out-of-memory errors, you can clear the cache simply by closing windows. (The layer editor has a button that lets you clear the cache without closing its window.) In this version of Audio Compositor, there is a seperate cache for each open window and another for the realtime engine. This is one reason why using the realtime engine for all audio output is more memory-efficient than playing sounds from the individual editors. A future version of Audio Compositor will improve on this scheme.

The term "cache" is used here somewhat loosely. In reality, some of what is cached will be in physical memory at any given moment, while the rest sits in your computer's pagefile. Allowing samples to be paged in this way may sound pointless--after all, once a sample has been bumped from physical memory, it is as easy to read it again from its original file as from the pagefile. But in practice this is not quite true: your virtual memory subsystem can quickly retrieve the small portion of the sample that is needed at a particular moment, while reloading the entire WAV file would be much slower. Thus, where MIDI file rendering is concerned, raising the cache limit may allow Audio Compositor to run significantly faster. However, there is no need to raise it above the total size of your sample collection.

If the caching limit is set too high, the MIDI file renderer may try to allocate memory that is not available even in your pagefile, and issue an out-of-memory error. Under Windows 95 with a dynamically allocated pagefile, the caching limit should be set safely under the amount of free space available on the disk where your pagefile is located. If Audio Compositor is writing to this same disk, take that into consideration too, or your output file may butt heads with the pagefile as they both grow.

The size of your cache, and the pagefile space available, are displayed continuously in the MIDI file renderer's status bar when it is running.

If you are very short on pagefile space, you may find the the only way to manage a large score is to process it one or two tracks at a time. See the Tracks section of the Job Setup dialog.