Audio Compositor

Introductory Topics

Using the Object Editor (Acoe.exe)

Running Audio Compositor (Ac.exe)

Introduction: What Audio Compositor Is

Audio Compositor is software for creating digital recordings of music scores using a personal computer. The input is a standard MIDI file; the output is a Windows WAV file. Sounds are derived from user-provided samples, which are also in WAV format. Audio Compositor is designed to overcome some of the limitations of the traditional small MIDI studio. For example, it works comfortably with extraordinarily large collections of input samples, and it can produce an almost unlimited number of simultaneous voices.

As you might expect, Audio Compositor has some limitations of its own. First of all, it does not come with a ready-made sound library--you must do your own collecting. This in turn says a lot about the program's intended audience, which will not include very casual users.

A second limitation is that Audio Compositor does not necessarily work in real time. A dense score can easily take longer to process than its musical duration--sometimes far longer. In this case you will not be able to listen to the output file as it is being written (actually, you can listen, but there will be gaps in the sound). Real-time audio is provided mainly as a curiosity, though as faster processors become affordable it may prove increasingly useful.

Audio Compositor is really a pair of programs. **Ac.exe** is the engine which actually produces the audio output. **Acoe.exe** is Audio Compositor's "Object Editor," a graphical environment in which to edit samples, layers, patches, and the other objects that make up Audio Compositor's musical architecture.

Why Use Audio Compositor?

Music production on general-purpose computers is nothing new (indeed it considerably predates the commercial synthesizer), and a variety of software for this purpose can be found today--perhaps the best known is Csound. The spirit of most such software is to encourage experimentation with synthesis techniques that aren't, and in many cases 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. In its present form at least, it's designed for musicians who're interested in testing the limits of traditional "wavetable" technology. To that end, Audio Compositor offers the following:

> Sample storage is limited not by available memory, but by available disk space. Sample memory layouts that would be unthinkable in an ordinary MIDI module become quite practical.
>
> The available polyphony (in the MIDI-jargon sense of the term) is unlimited.
>
> Most of the small MIDI studio's usual sources of noise and distortion are eliminated. MIDI bottlenecking at the serial interface is also a non-issue, and the timing of events in the output file is accurate to the nearest sample frame.
>
> It's very inexpensive.

In short, if you have sampled every half-step of your neighbor's crumhorn in stereo at 48 kHz, at six different dynamic levels, and can't fit the whole mess into your keyboard at one time, Audio Compositor will accommodate you.

It is also possible to view Audio Compositor more generally as a utility for assembling any kind of audio material that can be described by a MIDI file. You might find uses for it, even non-musical ones, that I haven't thought of.

Functional Overview

Audio Compositor produces its output by pasting sounds from your sample collection into a WAV file, using your MIDI file as a blueprint. In the process, it performs most of the functions that a synthesizer does, such as pitch-shifting, looping, and applying amplitude envelopes. The parameters governing these functions are stored in a pair of auxiliary files called the layer file and the patch file, using a structure and vocabulary which, for the most part, will be familiar to anyone who has spent a little time programming a synthesizer.

Audio Compositor's MIDI implementation 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.

The internal architecture is quite simple. Samples are mapped to the keyboard to form basic voices called *layers*. A *patch* may contain any number of layers, and the layers may sound simultaneously or be separated by velocity-based crossfades. At the time a layer is assigned to a patch, it also gets an amplitude envelope, controller routings, etc.

The 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!) Every WAV file is treated as an individual sample, and you begin simply by telling Audio Compositor where to find each one. If the sample must be looped, or if it will ever need to be pitch-shifted (this probably includes most of your samples), then you will also need to edit it using Audio Compositor's waveform editor, which imbeds information about the sample's loop points and pitch into the WAV file itself.

Samples may have been recorded at any sampling rate (though it's generally better 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 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.

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 typically 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. Several 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. It will also run on Windows 3.1 or Windows for Workgroups if you have the Win32s extensions, but real-time playback will not be available. Testing on systems other than Win95 has not been thorough.

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 unless an external FPU is present. The program is math-intensive and should respond well to faster CPUs.

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 would 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.

A sound card is a practical necessity, though the program will run without it.

Packing List and Installation

Audio Compositor consists of two executables: the Object Editor (Acoe.exe) and Audio Compositor itself (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\Acoe.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:\Software\Ac
```

then it will look for the guitar samples initially in

```
h:\Software\Ac\Samples\Guitar
```

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

Disclaimer

Audio Compositor is, and perhaps always will be, experimental software. The author makes no warranty of any kind as to its suitability for any purpose. You may use it only if you agree to do so at your own risk.

Audio Compositor Object Editor

The Object Editor, **Acoe.exe**, is used to prepare samples, layers, and patches for use with Audio Compositor.

The actual work of turning a MIDI file into digital audio is done by a separate program, **Ac.exe**.

Overview of the Object Editor

The Object Editor is used to prepare the three types of files needed by Audio Compositor:

Sample files, which are Windows .WAV files containing individual instrument samples.

The **layer** file, which keeps track of the samples on your hard disk(s) and maps them to notes on the MIDI keyboard. A keyboard mapping, together with some other parameters, defines a layer. Layers are the middle-level objects in Audio Compositor.

Patch files, which describe sets of up to 128 patches each. 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.

The Object Editor is just an empty window when you first start it up. Pull down the **File** menu to begin, and choose one of the three Open commands (Wave, Layer, Patch). You can also choose **New** to create a new, empty patch or wave file. (An empty .WAV file is only useful if you plan to use the **Synthesize** function.)

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.

Audio Compositor and .WAV Files

Audio Compositor uses samples stored in standard Windows .WAV files. Each file will normally contain a single instrument sound. When you set a sample's loop points and pitch using the Wave Editor, the loop and pitch information is imbedded in the .WAV file when you save it.

This loop and pitch information is written in a format that is only meaningful to Audio Compositor, but it won't interfere if you open the same file with another editor. In fact, it's not uncommon for audio editors to insert "private" data in .WAV files, as their format explicitly provides a safe way to do so. However, some editors discard chunks of data that they don't recognize (the current version of Audio Compositor, I'm afraid, is itself guilty of this). This means that if you set loop points and pitch in a .WAV file using the Object Editor, and then edit and save the same file using a different editor, the loop and pitch settings could be lost. You will need to experiment to find out whether a given editor works this way.

Some samples do not require either loop points or pitch information--an example would be an unpitched percussion sound. Samples like this will work just fine in Audio Compositor without being edited by the Wave Editor.

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.

Navigating the Wave File

When you open a .WAV file in the Object Editor, you will see the audio 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, ACOE displays them together on the same axis, in different colors. The left channel is always shown in black. (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.

 **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. The markers, when set, are indicated by solid green and red lines in both the upper and lower views. 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 dashed green and red lines, 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 single-sample increments. Conversely, you can back off for a wider view when you want to make large adjustments quickly.

If you click **Play With Loop** and leave the sound playing, you will hear the loop change as you edit it.

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.

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 has another technique for reduced looping. A normal MIDI instrument never knows 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 the 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 used in a production run. Changing the pitch setting will have no effect on the sound you hear when the sample is played 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.

See also the discussion of loop pitch and the **Compensate** check box, in Tuning Single-Cycle Loops.

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

Tuning Single-Cycle Loops

This topic is pretty tedious; you may wish to ignore it until a need arises.

Very short loops that correspond to a single cycle of an instrument's pitch are sometimes useful, but they present a special tuning problem, especially at higher pitches. The length of a cycle is apt not to correspond exactly to an integer number of samples--that is, you may find that the ideal length for your single-cycle loop is, say, 18.5 samples. In this case, since loop points in Audio Compositor are always whole numbers, you will have to choose a loop length of either 18 or 19. In the first instance the pitch heard at playback will shift upward when the loop is reached, in the second it will shift downward. In effect, a very short loop has a pitch of its own which conceals the pitch of the sample. The error in this pitch with respect the sample's "nominal" pitch is shown, in cents, as the **Loop pitch offset** in the Pitch Dialog. If the loop is nowhere near single-cycle length, the offset will register as "n/a."

As an example, consider a glockenspiel note pitched at exactly F#7 (~2960 Hz) and sampled at 44100 Hz. A single cycle of this sound will occupy 44100 / 2960 samples (about 14.9). If you try to make a single-cycle loop, you will end up setting it to a length of 15 samples. The "artificial pitch" of this loop is 44100 / 15 or 2940 Hz, which is about 12 cents flat, and when you listen to the sample you will hear the pitch fall slightly when playback reaches the loop stage. If the sample's pitch is set accurately, the pitch offset will be displayed as -12.

Under these conditions, Audio Compositor can solve the pitch-change problem if you will:

> Be sure you have set the pitch accurately for the overall sample. One way is to select a three-cycle region that includes your loop, and run Guess Pitch from the **Edit** menu.
>
> Choose the loop length that sounds best *disregarding* the pitch-change problem.
>
> Check the **Compensate** box in the Pitch Dialog.

Audio Compositor's wave editor does not do pitch-shifting on the fly, so setting **Compensate** does not make any difference in the sound you will hear when listening in the Wave Editor. But it sets a flag in the sample file that tells Audio Compositor to adjust the loop pitch during a production run. Inside the loop points, Audio Compositor will adjust its resampling process so that the frequency of the loop matches the pitch you have set for the overall sample. This is why it's essential that you set the pitch accurately.

If you check the **Compensate** box for a sample whose loop is not a single cycle, you will get extremely unpleasant results. The box is grayed out when it's obvious from the sample's pitch and loop length that a single-cycle loop is not the objective.

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
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 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.

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 separation 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, while raising the second value (and reducing the first as necessary) will introduce some right-channel content into the left channel. Applying this operation 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 (that is, during a production run). This function performs a "hardcoded" crossfade, permanently modifying your audio data. Obviously this is not an ideal solution, but as some samples 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 at 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 then 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 options are described under Resampling Algorithms.

Note that Audio Compositor never requires you to resample manually, since it can accept input material at any sample rate. This function is provided partly because it allows you to preview the effect of Audio Compositor's resampling algorithms on an isolated sound. It can also come in handy if you need to deal with sounds recorded at non-standard sampling rates. At least one high-priced sound card that I'm aware of is happy to play a file recorded at, say, 47000 Hz, but the result is severely distorted. In this case, resampling to 44100 is the only way to find out what such a file will really sound like when realized by Audio Compositor.

Resample

Calls up the Change Playback Rate dialog, which allows you to change the nominal playback 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 is 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 usually works well. It estimates the pitch of a small section of the sample; you must first locate this section by selecting a region of the sample display. The **Guess Pitch** item will be grayed out on the menu unless a region is selected. The region you select should include at least two, but less than four, full cycles of the sample's actual pitch. Without this hint, **Guess Pitch** turns quite stupid and will produce bizarre results.

The result of the function is a message like the following--answering "Yes" sets the pitch of the sample to the proposed value:

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

It is up to you to sanity-check this. The estimate is likely to be either quite accurate or wildly mistaken--in the latter case select a slightly different region and try again.

Since **Guess Pitch** operates on such a small section of the file, it doesn't work well for samples whose pitch is unstable. Samples with vibrato will give varying results depending on where your selection starts.

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 maps 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, though this requires some patience as a key will not speak immediately the first time it is pressed.

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 self-explanatory **Add**, **Edit**, and **Delete** buttons. Double-clicking on the name of a sample is a shortcut for the **Edit** button.

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

File Name

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.

Name

To make the layer editor easier on the eye, each sample is given a short name in addition to its full file specification. 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 observe this convention for you if 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.

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 use **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

You can "play" the keyboard, in a rather disappointing way, by clicking on it with the right mouse button. The first time you play a particular key, you will have to hold the mouse button down for a bit while its sample is loaded into memory and perhaps transposed. If you release the button before this process is complete, you won't hear the sample until you right-click again on that key. As you will quickly discover, the on-screen keyboard is not meant to be musically useful, but it gives you a way to audition your sounds and split points as you assemble a layer.

Once you have played a particular note, it is cached in memory and will speak more promptly the next time you right-click on it. As you continue to play different notes on the keyboard, more samples 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 (specifically, this purges those samples that are pitch-shifted copies of other cached samples, since these can be recalculated without reading again from disk). To purge *all* of the cached samples, press **Clear All**.

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 few hundred samples, you will have invested a great deal of time in telling Audio Compositor where to find them all. For example, if you decided to move your entire collection of samples to a new disk drive, you could be faced with the rather mind-numbing task of editing every entry in the sample list to begin with a new drive letter.

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 the file.

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.

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 usually means WordPad on Win95).

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.

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"--reminiscent of the Windows Explorer--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 (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 inherit 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 potentially confusing but often convenient arrangement. The patch displayed is always the one selected in the tree view (if the item selected is not a patch, its parent patch is shown).

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 Patch Editor includes an "Audition" box whose controls allow you to listen to the current patch, one note at a time. The two sliders allow you to choose the note and its velocity. When you press **Play**, the sound will be assembled just as it would be during a production run, and sent to your sound card. This process may take some time to complete, depending on factors such as the number of layers in the patch.

You do *not* need to hold down the **Play** button--in fact it doesn't do anything until you release it. The length of the sound is fixed at three seconds (though it will be less if the patch is composed of short, unlooped samples).

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 in the Layer Parameters window. The layer parameters are interpreted as if all MIDI controllers, aftertouch, etc. were set to zero--so unfortunately you cannot audition the effects of MIDI controller routings. Enhancements in this area are on the Audio Compositor "wish list."

Per-Layer Parameters

The Per-Layer Parameters dialog appears in the lower-right part of the Patch Editor. This is where most of the interesting synth-like parameters in Audio Compositor are set.

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 in 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 display, 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). 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 the 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 exactly 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. The pitch wheel is also displayed as ranging from 0 to 127, though its full 14-bit resolution is used internally.

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).

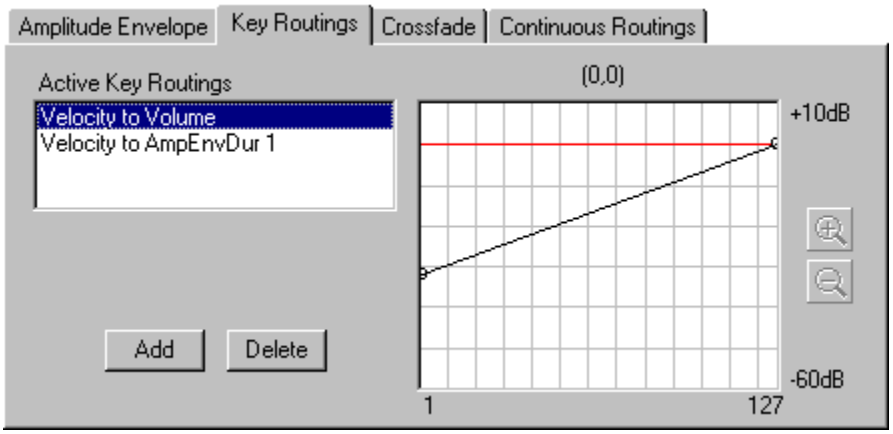
The y-axis units likewise depend on the destination of the routing. If the destination is either amplitude, or the level of a stage of the amplitude envelope, then the y-axis shows a dB value which is added to the destination. If the destination is the length of an envelope segment, the y-axis shows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

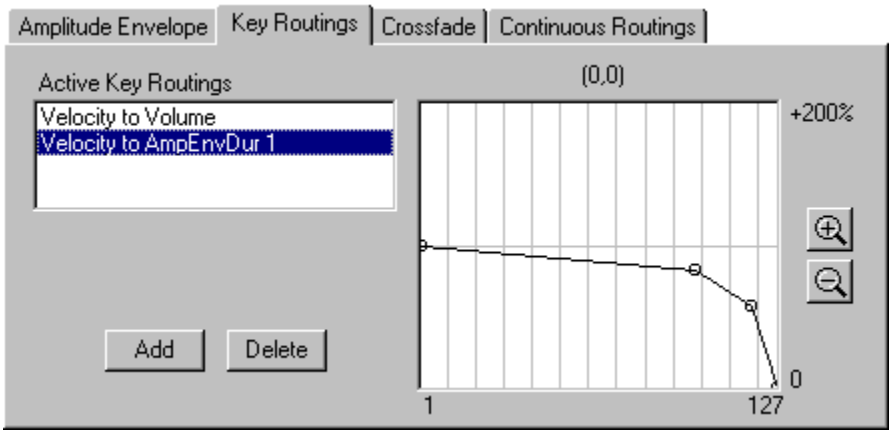
See also Routing Examples.

Routing Examples

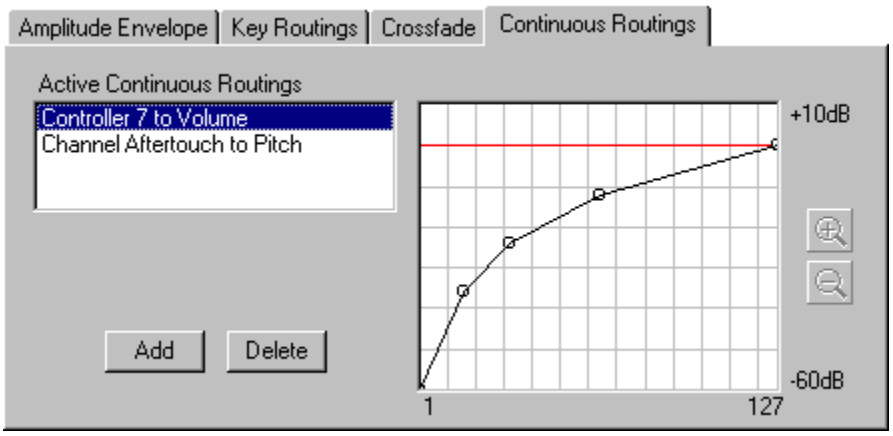
Virtually every layer will route velocity to volume (amplitude) as in this first example. If you want Audio Compositor to mimic the "feel" of a particular MIDI keyboard, you might find that an upwardly-convex shape is more accurate than the straight line shown here. Note also that while commercial keyboards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



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.

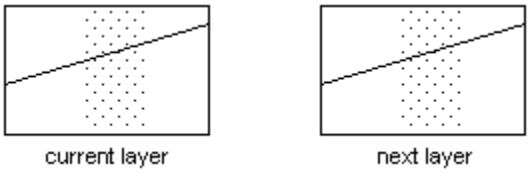
Width

> 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).

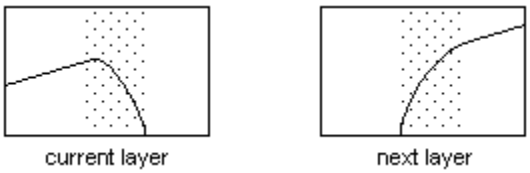
Type

> 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 change them to something 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. You will probably want to take care to cross the x-axis exactly at a value of 64.

Object Editor Preferences

The Preferences dialog (**Edit | Preferences**) is used to set a few global parameters for the Object Editor:

Sample Path Root

If this is set to a valid directory, Audio Compositor will change to it when you choose **Open Wave File** from the **File** menu. If you keep your samples in a different directory tree from Audio Compositor itself, this can save you some steps. This option is merely a convenience and has no bearing on Audio Compositor's other operations.

Minimum Sample Loop

This affects playback in the WAV editor only. When playing a looped sample, 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 "hardware" looping exclusively would defeat the WAV 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. (Since there are a number of variables at work, try decreasing it as well.) Some soundcards may not be able to do their own looping; in this case try using an absurdly high value (say, a million) to prevent Audio Compositor from ever requesting it.

Audio Compositor Production Engine

Audio Compositor (**Ac.exe**) realizes standard MIDI files as digital audio, deriving its sounds from instrument samples on your hard disk. The samples, as well as Audio Compositor's output, are in .WAV format.

The samples that Audio Compositor uses must first be prepared, and organized into MIDI patches, using Audio Compositor's Object Editor (**Acoe.exe**).

Overview: Producing Music with Audio Compositor

Ac.exe is the "engine" which actually draws together your MIDI file and the samples, layers, and patches you've created using the Object Editor, to produce a digital audio realization of a piece of music. A session with Audio Compositor goes something like this:

Start Audio Compositor (Ac.exe), and load the MIDI file you wish to realize.

Click the **Job Setup** button. This brings up a dialog in which you define the characteristics of the job, such as the name 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. The output is a standard WAV file that you can listen to with the Windows Media Player or any audio editor. Depending on a variety of factors, you may also be able to listen to the program's output while it is running.

When Audio Compositor finishes the job, you can do any of the following:

Shut down Audio Compositor (**File | Exit**).

Set up parameters for another run, using either the same or a different output file.

Open a different MIDI file and set up for another run (presumably, but not necessarily, with a different output file).

See also Audio Compositor Toolbar Buttons

Audio Compositor Toolbar Buttons

 **Open File**

Invokes the normal Windows "Open File" function to open a MIDI file. (Recently-used files can be opened more conveniently from the **File** menu.)

 **Job Setup**

Calls up the job-setup dialogs, which define what will happen when you press **Go**. See Setting Up a Production Run.

 **Go**

Starts the 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, and replaces 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.

This button is disabled if you are running on Windows 3.1 or 3.11, because the necessary threads support is not in the Win32s extensions.

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. The first track is presumed to be a "control" track with information such as time signature and tempo changes rather than note messages. (Your sequencer probably observes this convention whether you're aware of it or not.)

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; others like All Notes Off simply don't make sense in a software-only 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 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 separate 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 submitting your MIDI file to 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.

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

Opening the MIDI file

Unlike the Object Editor, Audio Compositor itself only knows how to open one kind of file--a standard MIDI file as described on the previous page. Use the regular **Open** command on the **File** menu. There is no **Close** or **Save** command, because Audio Compositor never modifies your MIDI file. If you want to replace the currently loaded MIDI file with a different one, simply choose **Open** again.

When you open a MIDI file, its tracks are displayed as horizontal divisions of the program's main window. Nothing happens if you click your mouse in this region, which is nothing more than a big progress indicator. Its purpose is to show you what is going on while Audio Compositor is working.

Setting Up a Production Run

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 this shortcut button on the toolbar: This calls up a tabbed dialog with several sections:

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, since otherwise Audio Compositor will be exercising your computer to no purpose. Declining an output file can speed up real-time output considerably.

Patch File

The patch file to be used in this run.

Layer File

The layer file to be used in this run. This will default to **Ac.lay** in Audio Compositor's home directory, and should only be changed if you have 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 "Process Selected Tracks" is checked, then Audio Compositor will make one pass through the file, processing only the highlighted tracks.

For relatively simple scores it may be practical to select all of the tracks, and have Audio Compositor produce 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 separately 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 process a subset of tracks and then, without necessarily exiting from Audio Compositor, select some or all of the remaining tracks and process 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, processing 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 process them together in order to take advantage of Audio Compositor's caching.

Range

By default, Audio Compositor will process 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 are sending the output of several separate 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, but this is a very small value. 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. 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.

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 output. The effects will vary between computer systems.

Granularity

This parameter tells Audio Compositor how often to recalculate a sound's amplitude when it may be changing over time in response to an amplitude envelope. (I believe it's somewhat 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 considerably. 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--it is combined with the "level" settings found throughout the Object Editor 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 (it's on the wish list).

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 processing 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.

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 any of three 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.

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. My own feeling is that it is only needed for particular sounds, and empirically speaking there are even 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 run.

Drop/Hold Sample

This is the fastest possible resampling method. No interpolation is done; samples are simply omitted (when downsampling) or repeated (when upsampling) to create a crude approximation of the original signal. The sound is uniformly appalling. The only reason to use this method is to create a rough draft of something when you are in a great hurry, before doing it over again using one of the other options.

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. The **Maximum Sample Cache** parameter (specified on the **Constants** page) limits its 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 realized.

The term "cache" is used here somewhat loosely--in reality, some of it will be in physical memory at any given moment, while the rest sits in your computer's pagefile. Paging out samples 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. So raising the cache limit may allow Audio Compositor to run a little faster.

If the caching limit is set too high, Audio Compositor may try to allocate memory that is not available even in your pagefile, and you'll receive 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.

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.