

## What is Quack

Quack is a Win95 application which can be used to generate a wide variety of electronic sound effects. The sounds produced by Quack can be stored as WAV files, and may be used by any other applications which accept the WAV format (including Sound Recorder and Media Player).

Quack is modelled on an analog “patch board” synthesiser. It provides a wide variety of audio components - such as oscillators, filters, modulators, envelope generators, noise sources - which can be connected together to produce complex sounds.

Quack can also read in existing WAV files (for example, sounds sampled using Sound Recorder), and incorporate them into sound effects.

For a step by step guide to producing a sound file, read the Overview.

## Overview

There are a number of stages involved in creating a sound with Quack.

First, you must design the sound you want to create. Quack provides a large number of Components which can generate or process sounds in different ways (see the List of Components in the table of contents). These components may be connected together to form a more complex sound. The map of connected components is called a Patch Map.

Once you have created the patch map, you will probably want to save it to file, in case you want to use it again.

Finally, you can create the patch map to produce a sound file in WAV format. And if you wish, you can play the WAV file from within Quack to check it.

Continue with this browse sequence to learn more about these stages, starting with an example of a patch map.

## Patch Maps

This topic illustrates the concept of a patch map using a simple example. After reading this, continue with this browse sequence to learn more about the procedures involved.

This example simulates a very simple percussion instrument - a wood block being hit with a drumstick.

The first stage is to design the sound. We might decide that the basic sound of the wood being struck is a mixture of a pure tone and a noise signal. The sound is very percussive and short in duration, so we will need a tight envelope to contain the sound.

In all, we will need to create four components:

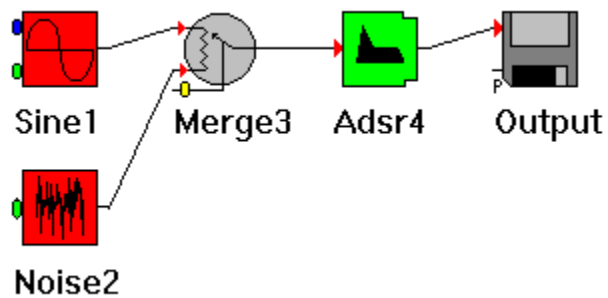
- o A Sinewave Generator to produce the pure tone. This should be set up with frequency 1000, volume 1, offset 0.
- o A Noise Generator to produce the noise component of the sound. This should be set up with a volume of 1.
- o A Merger to mix the sinewave and noise signals. This should be set so that the two signals are mixed equally (the default setting).
- o An ADSR Envelope Generator to generate the envelope for the sound. Use an attack time of 0.01, a decay time of 0.02, a release time of 0. Set the attack level to 1.0 and the sustain level 0. This gives a simple Attack/Decay envelope.

The patch map also uses the Output component. This component is responsible for actually generating the sound file. You don't need to create an output component, because it is created automatically with every new patch map. You will need to edit the output component. Change the duration parameter to 0.1, because we only need to generate a short file for such a short sound.

Next we must create connections as follows:

- o Connect the ADSR Envelope Generator *Output* to the Output Component *Source*.
- o Connect the Merger *Output* to the ADSR Envelope Generator *Source*.
- o Connect the Sinewave Generator *Output* to the Merger *Source A*.
- o Connect the Noise Generator *Output* to the Merger *Source B*.

The patch map displayed should look like:



Notice that several inputs are not connected. For example, the Sinewave Generator *Frequency* input is not connected. This simply means that the frequency is not controlled by an external source, it remains constant at 1000 (the value we set when we created the component).

If you don't want to go through and create this patch map for yourself, open the supplied example file **block.pm**. For other examples, see the [List of Examples](#).

## Diagram Notation

All the component icons follow a common notation. It is useful to learn this, because it helps to clarify the diagrams.

The shape and colour of a component indicates its basic functional type:



Components which produce audio signals



Components which combine 2 or more signals



Components which generate envelopes



Components which filter signals

Each input signal on a component is marked with a symbol to indicate the type of signal:



An audio signal input



A signal which controls the volume of the audio signal



A signal which controls the frequency of a signal



Any other general control

Take the example of a Comb Filter:



The basic shape of this component reminds us that it is a type of filter (ie it modifies the signal passing through it). The description of the comb filter tells us that it has 3 inputs - the signal, the filter frequency, and a filter strength control (do not worry too much about what a comb filter does, at this stage - this is just for illustration). The notation acts as a reminder that the first input (red arrow) is the main signal, the next (blue circle) is the frequency, and the last (yellow circle) is the strength.

The output component is responsible for generating the sound file. Every patch map must have one (and only one) output component. The output component is present in the patch map when it is created - you cannot delete this component, or create new versions of it. You can modify the parameters of this component.

## Using Components

A patch map is created by connecting components together.

This section describes how to create, edit and delete components in a patch map.

After reading this topic, continue the browse sequence to learn how to connect the components to produce a patch map.

### ***Creating a new component***

#### **With the mouse:**

Use the **Components** floating palette to easily add new components.

- If the **Components** floating palette is not visible, choose **View|Components** on the main menu.
- Click the required component in the **Components** palette.
- A dialog box will appear which allows you to set up the parameters of the new component - the dialog depends on what type of component you have created. See the List of components in the contents page.

#### **With the keyboard:**

- Choose **New** from the **Components** menu. A second menu will appear which lists all the available components. Choose the one you require.
- A dialog box will appear which allows you to set up the parameters of the new component - the dialog depends on what type of component you have created. See the List of components in the contents page.

The new component will automatically be given a name (for example, a modulator will be called **mod 5** or something similar, where the number is unique). It will be displayed in the patch map, initially unconnected to any other components.

### ***Editing a component***

To change the parameters of an existing component. You can do this whenever you need to (before or after you have connected the component).

#### **With the mouse:**

Double click on the component, and its parameter dialog will appear.

Alternatively, click on the component with the right mouse button and choose **Edit** from the popup menu which appears.

**With the keyboard:**

- Choose **Edit** from the **Components** menu.
- Choose the component you wish to edit from the list box which appears.
- A dialog will appear allowing you to change the parameters.

***Deleting a component***

**With the mouse:**

- Click the component with the right mouse button.
- Choose **Delete** from the popup menu which appears.
- The component and any connections will be deleted.

**With the keyboard:**

- Choose **Delete** from the **Components** menu.
- Choose the component you wish to delete from the list box which appears.
- Note that you cannot delete a component which is still connected to something else - you must delete all its connections first.

## Using Connections

A patch map is created by connecting components together.

This section describes how to create and delete connections in a patch map.

After reading this topic, continue the browse sequence to learn how to save your completed patch map, and how to generate a sound file using the patch map.

### ***Creating a new connection***

#### **With the mouse:**

In Quack, a connection is always made from the *output* of one component to an *input* of another component (not the other way around).

- Position the cursor over the first component.
- Hold down the left mouse button, and drag the cursor to the second component. The cursor will change to a crosshair.
- Position the cursor over the input of a second component and release the button. The cursor will change to a target.

#### **With the keyboard:**

- Choose **New** from the **Connections** menu.
- A dialog box will appear so that you can create a connection.

A connection can be made output node of one component and the input node of another:

- o In the **From** box, choose the component and output node which is the source of the signal. (In the current implementation, all components have one output node, called **output**, so you don't really have much choice about the output node).
- o In the **To** box, choose the component and input node which is the destination of the signal

Connections are strictly one to one. No node can ever be connected to more than one other node. A warning will be issued if you try to do this.

### ***Deleting a connection***

#### **With the mouse:**

- Position the cursor over the component whose output you want to disconnect.
- Click the right mouse button.
- Choose **disconnect** from the popup menu

**With the keyboard:**

- Choose **Delete** from the **Connections** menu.
- Choose the connection you wish to delete from the list box which appears.

## Saving and restoring patch maps

Use the standard **File** menu commands **Save**, **Save As**, **Open** and **New** to save and restore patch map files. You can only work on one patch map at a time when using Quack.


Patch maps are usually stored with the filename extension *.pm*. For example **patch.pm**

Remember that the patch map file is not a sound file. The patch map describes how to create a particular sound, but it does not contain the sound itself. Continue with this browse sequence to learn how to create a sound file from the patch map.

Also remember that if you use the WAV Sample component to read samples into your sounds, the sample files are **NOT** stored as part of the patch map. They must be backed up separately if they are important.

## Creating a sound file

Before creating a sound file, you must create a patch map as described in the Overview and subsequent topics.

Either click the  **create** button on the toolbar, or select **Sound|Create** from the menu.

**TIP:** Use **Set Soundfile** to speed up your work. Choose the **Sound|Set Soundfile** to define a *working file*. After that, the **create** option will automatically write to the working file, without bringing up the **File Save As** dialog. (To turn this option off, just click **Sound|Set Soundfile** again).

When you hit the **OK** button, the sound file will be generated. The amount of time taken to create a sound file depends on the length of the sound, the complexity of the patch map, and of course the power of your computer. As a guide, a 486 DX/2 66 should process a moderately complex patch map in almost real time (i.e. it will take about 10 seconds to produce a sound file which takes 10 seconds to play - but this is only a rough guide).

After creating the file, Quack will report the peak amplitude of the signal stored to file. This amplitude should be somewhere between 0.5 and 1.0 for best results - if the signal is outside this range, you should change the *Volume* of the Output component as the report instructs. If you don't, the sound produced could be distorted, and will certainly not be as good as it could be.

The file produced will always be in the WAV format. You can choose the sample rate, bits per sample, stereo/mono mode, and volume by editing the Output component parameters. See the technical note on Output file parameters for more details.

## Playing a sound file


You can play a WAV file from within Quack Sound Effects Studio.

Either click the  **play** button on the toolbar, or select **Sound|Play** from the menu.

**TIP:** Use **Set Soundfile** to speed up your work. Choose the **Sound|Set Soundfile** to define a *working file*. After that, the **play** option will automatically play the working file, without bringing up the **File Open** dialog. (To turn this option off, just click **Sound|Set Soundfile** again).

## Working with larger Patch Maps

As you become more familiar with Quack, and start to generate more complex patch maps, you may want to use a *condensed* view. This allows you to see more of the patch map on your screen.

Either click the  **layout** button on the toolbar, or select **View|Condensed** from the menu.

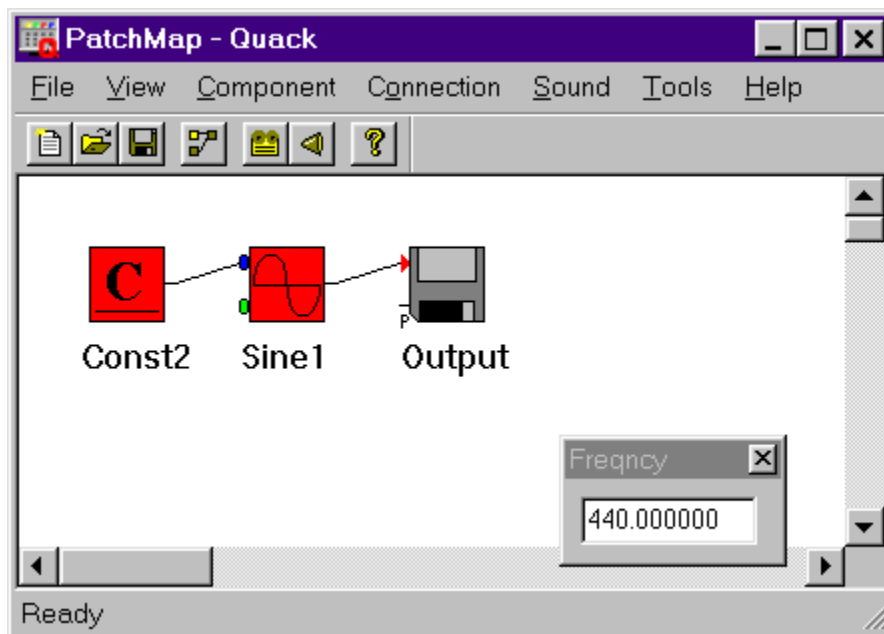
## Tweak Windows

As you get used to using QUACK, you may find that you need to play around with different parameters in your patch map until you get just the sound you require.

Continually opening and closing the edit dialogs for different components can become tedious in these cases, so QUACK provides a method to attach “Tweak” windows to key parameters in the patch map. These little floating windows provide an entry field which is always open for you to type in a new value for the parameter it represents.

Tweak windows actually form part of the Constant component. To create one, simply create a new Constant component, and select the Tweak option in its dialog box. You should also type in a name, which will appear as the title of the tweak window. Then, simply attach the Constant to whichever input you wish to control. The value you type into the Tweak window will directly control the value at the output of the Constant.

Here is a simple example of a tweak window controlling the frequency of a sine wave:



There are other examples in the Examples folder.

## Tools Menu

There are many excellent freeware, shareware and commercial audio applications around which can be used in conjunction with QUACK to further enhance the sounds you can create. To make life easier, QUACK has a Tools menu to which you can add your favourite:

- WAV Player
- WAV Recorder
- WAV Editor
- Spectrum Analyser

Use the Tools|Properties menu option to define the path of whichever program you want to use for each function. You can use the same program for more than one of these functions, of course, or you can leave some of the fields blank if you do not need them.

We recommend that you use the built-in WAV Player (the Sound|Play option) while you are experimenting, because it is more fully integrated and therefore quicker to use.



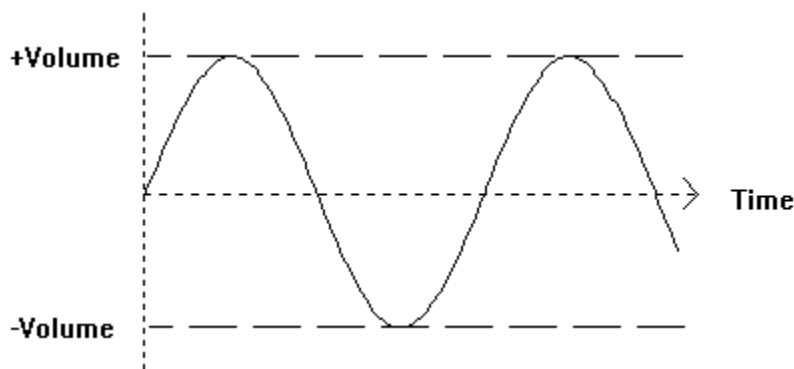
## Sine Wave Generator

Produces a sine wave output which may be used to produce a pure tone, or as an envelope to produce vibrato or tremolo effects.

### Parameters

The following parameters may be set in the Sine Wave Generator dialog box:

- *Frequency* - the frequency in Hertz (or cycles/per second) of the tone generated.
- *Volume* - the peak volume of the signal generated.
- *Phase* - the initial phase of the waveform.
- *Offset* - see [Note on waveform offsets](#)



The signal oscillates between two extreme values (given by plus *Volume* and minus *Volume*). Initially, the wave starts at zero (by default).

The *Phase* may be altered so that the wave begins at a different point in its cycle. This takes a value of 0% to 100%, which advances the wave in time by a fraction of a complete cycle.

### Inputs

The Sine Wave Generator has inputs for *Frequency* and *Volume*. These may be connected to the outputs of other components. If they are connected, they totally override the values set in the dialog box.

Note that there is no input to control the *Offset*, see [Note on waveform offsets](#)

### Uses and Examples

Use this component to generate pure tones in the audible range. Choose *Frequency* of anywhere from 20 to 20,000. Set the *Volume* to about 1, and use an offset of 0.0

To generate tremolo or vibrato effects, use a low frequency (less than about 20), and use the *Offset* to create a suitable waveform (see [Note on waveform offsets](#)).



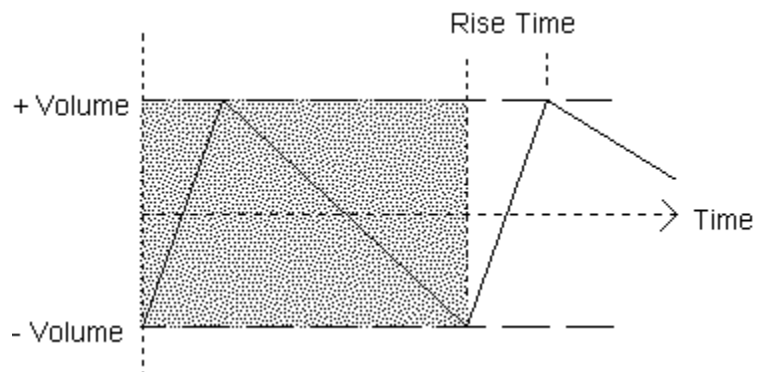
## Sawtooth Generator

Produces a sawtooth output which may be used to produce a tone, or as an envelope to produce vibrato or tremolo effects.

### Parameters

The following parameters may be set in the SawTooth Generator dialog box:

- o *Frequency* - the frequency in Hertz (or cycles/per second) of the tone generated.
- o *Volume* - the peak volume of the signal generated.
- o *Phase* - the initial phase of the waveform.
- o *Rise Time* - The rise time as a proportion of the total wave period (see below).
- o *Offset* - see Note on waveform offsets



The signal oscillates between two extreme values (given by plus *Volume* and minus *Volume*). Initially, the wave starts at the lowest value, as shown in the diagram. The shaded area represents one complete wave period.

The *Phase* may be altered so that the wave begins at a different point in its cycle. This takes a value of 0% to 100%, which advances the wave in time by a fraction of a complete cycle.

The *Rise Time* controls how much of the cycle is spent at the high value. It is a value between 0% and 100%. A *Rise Time* of 50% gives a true Sawtooth. The diagram shows a rise time of about 20% (the signal takes 20% of the time to rise, and 80% of the time to fall).

### Inputs

The Sawtooth Generator has inputs for *Frequency*, *Volume* and *Rise Time*. These may be connected to the outputs of other components. If they are connected, they totally override the

values set in the dialog box.

Note that the signal controlling *Rise Time* should have a value between 0 and 1 (corresponding to 0-100% in the dialog).

Note that there is no input to control the *Offset*, see Note on waveform offsets. There is no input to control the *Phase*.

## **Uses and Examples**

This component is similar to the Sine Wave Generator but has a different wave form, with higher frequency harmonics present, giving a more brassy sound.



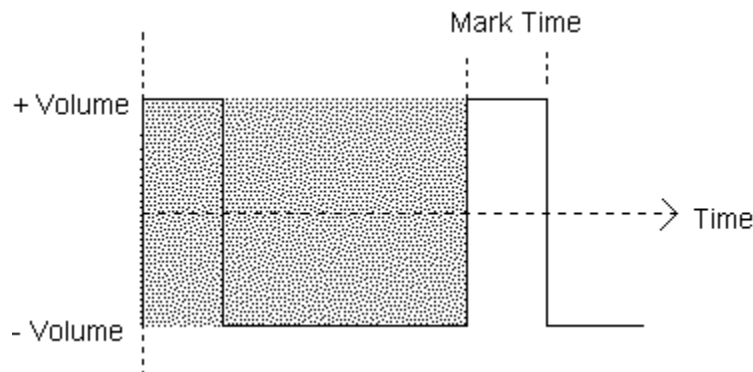
## Square Wave Generator

Produces a square wave output which may be used to produce a tone, or as an envelope to produce vibrato or tremolo effects. It may also be used to switch other signals on and off periodically.

### Parameters

The following parameters may be set in the Square Wave Generator dialog box:

- o *Frequency* - the frequency in Hertz (or cycles/per second) of the tone generated.
- o *Volume* - the peak volume of the signal generated.
- o *Phase* - the initial phase of the waveform.
- o *Mark Time* - The mark time as a proportion of the total wave period (see below).
- o *Offset* - see [Note on waveform offsets](#)



The signal oscillates between two extreme values (given by plus *Volume* and minus *Volume*). Initially, the wave starts at the high value, as shown in the diagram. The shaded area represents one complete wave period.

The *Phase* may be altered so that the wave begins at a different point in its cycle. This takes a value of 0% to 100%, which advances the wave in time by a fraction of a complete cycle.

The *Mark Time* controls how much of the cycle is spent at the high value. It is a value between 0% and 100%. A *Mark Time* of 50% gives a true Square Wave. The diagram shows a mark time of about 20% (the signal is high for 20% of the time, low for 80% of the time).

### Inputs

The Square Wave Generator has inputs for *Frequency*, *Volume* and *Mark Time*. These may be

connected to the outputs of other components. If they are connected, they totally override the values set in the dialog box.

Note that the signal controlling *Mark Time* should have a value between 0 and 1 (corresponding to 0-100% in the dialog).

Note that there is no input to control the *Offset*, see Note on waveform offsets. There is no input to control *Phase*.

## Uses and Examples

This component is similar to the Sine Wave Generator but has a different wave form, with higher frequency harmonics present, giving a more electronic sound.

An additional use for this component is as a gate to control other signals. To do this, set the *Volume* to 0.5, and use an *Offset* of 0.5. The result is a signal which switches between 1.0 and 0.0 during each cycle. If this signal is used to control the volume of another signal, it will effectively turn the signal on and off periodically.



## Noise Generator

Produces white noise output. White noise is a useful starting point for many sounds, especially after filtering.

### Parameters

The Noise Generator dialog box contains the following parameters:

- o *Volume* - the peak volume of the signal generated.
- o *Offset* - the average signal value.

The signal oscillates randomly between two extreme values (given by plus *Volume* and minus *Volume*). The signal produced has an approximately even frequency spectrum across the audible range.

If an offset is specified (other than zero), the signal oscillates randomly between *Offset+Volume* and *Offset-Volume*.

### Inputs

The Noise Generator has an input for *Volume*. These may be connected to the output of another component. If it is connected, it totally overrides the value set in the dialog box.

### Uses and Examples

The raw output from this component sounds like a continuous hiss. It may be used to add noise or hiss to percussive sounds like drum beats or cymbals. It can also be used to produce a “breathed” quality for wind instruments.



## Sweep Envelope Generator

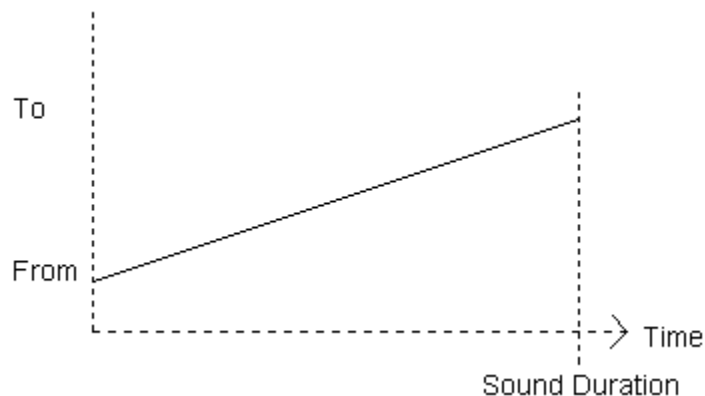
Generates a signal which gradually changes from an initial value to a final value throughout the duration of the sound. This can be used to control a volume or frequency which steadily increases or decreases throughout the duration of the sound.

### Parameters

The Sweep dialog box has two parameters:

- *From* - the initial value of the signal.
- *To* - the final value of the signal.

The signal changes from the initial value to the final value over the duration of the sound:



The signal may increase (as shown) or decrease with time, depending on whether the *To* value is greater than or less than the *From* value.

### Inputs

The Sweep has an input for *Source*. This may be connected to the output of another component. If it is connected, the output of the Sweep will be the sweep signal multiplied by the *Source* signal.

If the *Source* is not connected, the sweep signal will output unmodified (i.e. the default value of the disconnected *Source* is 1.0).

### Uses and Examples

Use the Sweep component to control any value which increases or decreases gradually with time - for example the volume or frequency of another signal.

The Sweep may also be used to control the *Pan* of the Output component. In this case the start and end values should be in the range 0.0 to 1.0. This produces a sound which pans from left to right (or vice versa) as it progresses.



## ADSR Envelope Generator

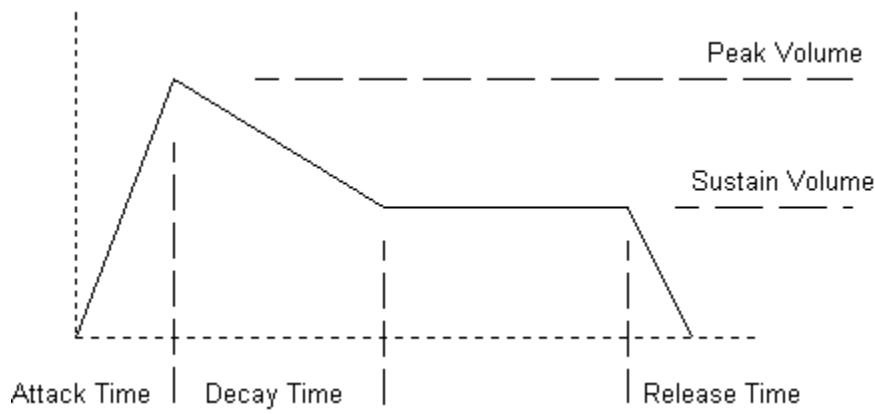
Generates an Attack/Decay/Sustain/Release (ADSR) envelope. This may be used to control the volume of a tone or other signal, to simulate a note being played on an instrument, etc.

### Parameters

The ADSR Envelope Generator dialog box has the following parameters:

- o *Attack*, *Decay* and *Release* times. See the diagram below.
- o *Peak* and *Sustain* volumes. See the diagram below.
- o *Curve* - controls the way the volume changes during the Attack, Decay and Release phases. See the technical note on [Envelope Curves](#).

The shape of the signal is as follows:



The envelope generated will always occupy the full duration of the file. The times of the Attack, Decay and Release phases are fixed, but the Sustain time is not. The Sustain time will be automatically adjusted so that the total time of all 4 phases is equal to the duration of the sound.

### Inputs

The ADSR has an input for *Source*. This may be connected to the output of another component. If it is connected, the output of the ADSR will be the envelope signal multiplied by the *Source* signal.

If the *Source* is not connected, the ADSR signal will output unmodified (i.e. the default value of

the disconnected *Source* is 1.0).

## Uses and Examples

The ADSR signal can be used to control the volume of a source of continuous sound (such as a wind instrument, or a string played with a bow). The volume initially increases as the sound starts (Attack), then dies away slightly (Decay), continues at a constant volume (Sustain), before finally falling away to zero (Release).

Careful choice of parameters can produce other useful envelopes:



**Attack/Decay** - this is useful for percussive sounds such as plucked strings, drumbeats etc. Set the *Sustain Volume* and *Release Time* to zero. Make sure that the sum of the Attack and Delay times is equal to the sound duration (set by the Output component).

**Attack/Decay/Release** - simulates a damped percussive sound. This envelope is set up much the same as an ADSR envelope, but the sum of the A, D and R phases is equal to the total sound duration, so that the Sustain phase is of zero length.

**Attack/Sustain/Release** - this produces a sound which is constant in volume, except at the start and end. This envelope can be applied to sampled sound sources, to ensure that the sound starts and ends smoothly, without any clicks or pops. Set the peak volume and sustain volume both to 1, and set the decay time to zero.

## Output Component

Every patch map has one (and only one) output component. This is the component which actually produces the output sound file. Whatever signal feeds the input of this component will end up in the sound file.

### Parameters

The Output Component dialog box has several parameters:

- *Volume* - controls the volume of the sound in the file. The easiest way to obtain the correct volume is to start with this parameter set to 1. When you **Run** the patch map to produce a sound file, Quack will detect if the signal is too weak or too strong, and advise you of a suitable value for *Volume*.
- *Duration* - controls the duration of the file, in seconds.
- *Samples Per Sec* and *Bits Per Sample* - see the technical note on [Output File Parameters](#).
- *Stereo Enable* and *Pan* - see the technical note on [Stereo Sound](#).

### Inputs

The Output component has a signal input (marked by an arrow) for the incoming signal. It also has a *Pan* input which controls the stereo position. This may be connected to the output of another component. If it is connected, it totally overrides the value set in the dialog box.

### Uses and Examples

Typically, the Output is simply fed by the signal generated by the patch map.

You may also wish to use the *Pan* input to move the sound around in space for special effects. For example, a sweep signal feeding the *Pan* input can make the sound move from left to right (or vice versa) over its duration.



## Modulator

Multiplies two signals together, so that the second signal acts as a variable “volume control” on the original signal.

## Parameters

The dialog box has the following parameter:

- o *Volume* - If only one signal is provided, on input A, this acts as a fixed multiplier on the original signal. For example, if volume is set to 2, the signal on A will be doubled in amplitude.

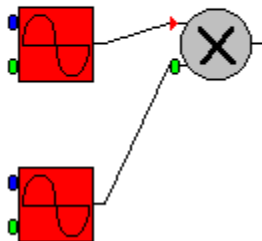
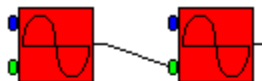
## Inputs

If signals are provided on Input A and Input B, the output is A times B. In that case, *volume* is ignored.

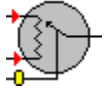
If a signal is provided on Input A only, the modulator acts as a fixed volume control, and its output is equal to A times the *volume*.

## Uses and Examples

Modulation is such an important function that most components have a modulator built in, in the form of a volume input. For example, the two patches below serve exactly the same function (they use one sine wave to control the volume of another). As you can see, the modulator is often unnecessary.



There may be exceptional cases where a modulator is needed. You might also find it useful when you are experimenting with new patch maps - it can be clearer, and easier to move things around, if you have a few redundant modulators. But beware - too many unnecessary components could increase the amount of time taken to generate the sound.



## Merger

A Merger component acts as a cross fader. It mixes two signals, by adding them. The proportion of the two signals may be fixed, or may vary with time.

### Parameters

The dialog box has one parameter:

- o A/B proportion - determines the percentage of the signal made up by signal A.

Eg:

1.0 - the output signal is equal to input signal A

0.0 - the output signal is equal to input signal B

0.5 - the output signal is 50% A, and the remaining 50% is B

0.3 - the output signal is 30% A, and 70% B

etc.

### Inputs

The Merger has inputs for *Signal A*, *Signal B*, and *A/B Proportion*. These may be connected to the outputs of other components. If they are connected, they totally override the values set in the dialog box.

The *Signal* inputs are marked by letters A and B.

The *Proportion* input is marked by a letter P.

If either of the inputs A or B is not connected, its signal will be set to zero.

### Uses and Examples

This component is used as a cross fader, to mix two other sounds.

It may be used to mix two signals in fixed proportions, or it may be used to cross fade dynamically between two sounds. In that case, the *A/B Proportion* should be controlled by some kind of envelope (for example a sweep or a sinewave).



## Sequencer

This component controls the start and end times of up to 4 different signals. It can also be used to automatically repeat input signals.

### Parameters

The dialog box has the following parameters for each signal (A, B, C, D):

- ☐ Start time
- ☐ End time
- ☐ Repeat

Each input signal plays from its start time until its end time. If repeat is enabled, the signal repeats continuously after the first playing. All four signals are controlled independently, and can overlap if required.

### Inputs

There are 4 signal inputs, for signals A to D.

### Uses and Examples

This component is useful wherever you wish to build a sound out of a number of separate sounds which all occur at different times. It can also be used simply to delay a signal, or to repeat a signal.

The best way to develop complex sounds like this is to build and test all the elements separately. When they are all working, add the sequencer and put the sounds all together.



## Comb Filter

A comb filter is a simple digital “notch” filter, which can be used to great effect in conjunction with noise or other complex signals.

A comb filter operates very simply. The input signal is fed through a delay and added to output signal. The output is given by:

$$\text{out}(t) = \text{in}(t) + \text{strength} * \text{in}(t - \text{delay})$$

The delay is  $1/\text{frequency}$ . Eg, if the required peak frequency is 100Hz, the delay would be 0.01 sec.

If we feed a 100Hz signal into the filter, we would find that the input signal and the delayed signal are exactly in phase. So adding one from the other tends to cancel them out. The same is true at 200, 300, 400Hz, etc. On the other hand, at 50Hz the signal and the delayed signal are in antiphase (one is the *negative* of the other). Adding them actually decreases the signal. The same is true at 150, 250, 350 etc.

The frequency response is a series of regularly spaced peaks and notches, hence the name of the comb filter.

A second variant of this filter uses the delayed *output* rather than the delayed input signal to feed back - this is called *Reverb* mode.

## Parameters

The dialog box has the following parameters:

- Frequency - this determines the frequency of the first notch.
- Strength - this determines how “deep” the notch is. Strength should normally be between 1 (full strength) and 0 (which turns the filter off altogether). Other values are allowed, but may give strange effects. **Note:** in Reverb mode *Strength* must be less than 1, because of the feedback effect, otherwise the signal could become unstable.
- Type - Simple or Reverb as described above, both do a similar thing, but with slightly different effects.

## Inputs

The component has a signal input, and inputs for the *frequency* and *strength*. If connected, the frequency and strength inputs completely override the parameters in the dialog box.

## Uses and Examples

The comb filter is most useful if its notch frequency varies with time. A classic use of a comb filter is in conjunction with a noise signal. to produce a variety of “whooshes” and jet aircraft sounds. Use a sweep or other envelope to sweep the comb frequency through several octaves. The Music modulator can be useful here.

Experiment with other complex signal too.

If you use the comb filter with simple waveforms, particularly square waves, you might not always get the effect you expect. For example, if you feed a 400Hz square wave into a 400Hz comb filter, you should (in theory) get nothing out - in fact, you might get a very narrow 400Hz spike waveform, due to the fact that wave is not continuous in time. In fact, the comb filter is best used with more complex waveforms.

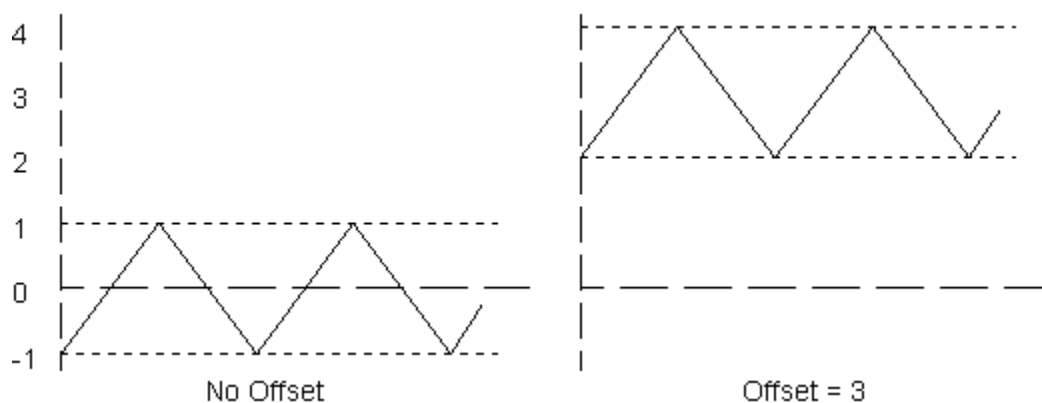
## List of Example Patch Maps

There are lots of example patch maps available in the Examples subdirectory. To make it easy to find your way around these examples, an index is provided in HTML format. Simply use any World Wide Web browser application to open up the file **index.htm** in the Examples subdirectory, and you will find all the examples indexed and fully explained.

## Using Offsets with Waveform Generators

By default, waveform generators produce a signal which oscillates about zero. For example, a Sawtooth with a volume of 1.0 will oscillate between +1.0 and -1.0. This type of waveform is useful for producing audible tones.

Sometimes it is useful to have a waveform which oscillates about some value other than zero. For convenience, all waveform generators allow you to define an *Offset*. When an offset is specified, the whole wave is shifted. For example, if an offset of 3.0 is applied to a sawtooth with a volume of 1.0, the wave will oscillate between +2.0 and +4.0:



This type of waveform can be used to generate effects such as tremolo and vibrato. Of course, other wave shapes can be used instead of the sawtooth.

For tremolo effects (where the volume of a sound oscillates slightly), use an offset of 1.0, and a low volume (a value of 0.2 is a good starting point, but this value can be varied to alter the effect). The frequency should be low (start with something between 1 and 10 cycles per second, but again this can be varied). This gives a signal which varies slowly, with an average value of about 1. Use this signal to control the volume of another sound.

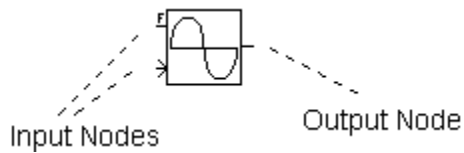
For vibrato effects (where the frequency of a sound oscillates slightly), use an offset equal to the basic frequency of the tone you want to create, and a volume equal to about 20% of the offset. For example, to produce a 1000Hz tone with vibrato, use a waveform generator with offset 1000, volume 200, and a frequency of 1 to 10. Use this output to drive the frequency input of a second waveform generator.

## Input and Output Nodes

Nodes are points where one component can be connected to control another component.

All components have one output node, called *Output*.

Most components have one or more input nodes. For example, the Sinewave Generator has two nodes called *Frequency* and *Volume*:



Nodes are indicated by short lines on the symbol for the component, which connect to other components in the patch map diagram.

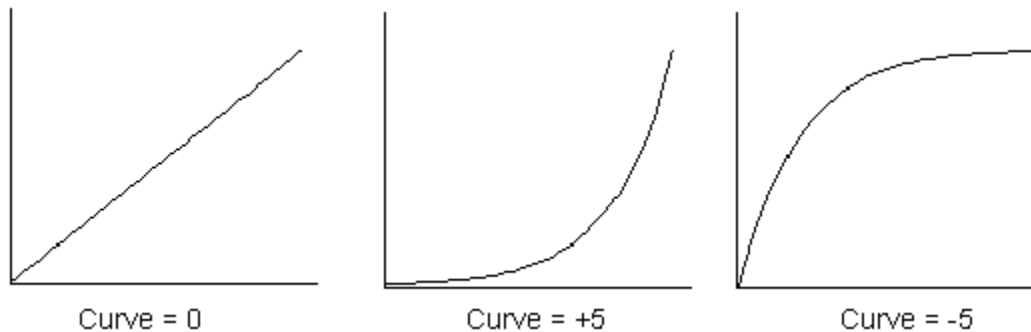
WAV format is a standard format for sound files, supported by many Windows applications including Sound Recorder and Media Player.

Quack uses a very simple version of the WAV format, and does not rely on any advanced features of the format. Files generated by Quack should therefore be compatible with as many other applications as possible.

## Envelope Curves

Functions which generate envelopes, such as the ADSR Envelope Generator generally specify a number of transitions, during which the sound level changes with time.

For example, the Attack phase of the ADSR envelope, starts at zero and rises to the peak volume, within a fixed amount of time (the attack time). However, many natural sounds do not change volume linearly with time. There are three possibilities, controlled by the *Curve* parameter. Notice that in all three cases, the curve starts in the same place and ends in the same place, it is just the route which varies.



In the first case, with a curve of zero, the rise (or fall) of volume is linear with time.

In the second case, the volume rises exponentially with time. That is, the change in volume is slow at first, but gradually becomes faster as the sound develops. This can be used to simulate sounds such as wind instruments. The value of +5 for curve is just an example. Any small positive value will give a similar result - larger values give a greater amount of curve.

In the final case, the volume rises as the inverse exponential of time. That is, the change in volume is rapid at first, but gradually becomes slower as the sound develops. This can be used to simulate sounds such as percussive instruments. The value of -5 for curve is just an example. Any small negative value will give a similar result - more negative values give a greater amount of curve.

For the mathematically minded, the basic function used to generate envelope transition curves is:

$$v = (1 - \exp(c*t))/(1 - \exp(c))$$

Where  $v$  is the volume,  $t$  is time,  $c$  is the curve parameter, and  $\exp$  is the exponential function. This formula takes  $v$  from 0 to 1 as  $t$  goes from 0 to 1 (this is scaled to produce real envelopes).

When  $c$  is positive, this is an exponential function. When  $c$  is negative this is an inverse exponential function. As  $c$  becomes very small, this function becomes more and more like a straight line,  $v = t$ .

Notice that a small problem occurs if we try to use this function with  $c = 0$ . Since  $\exp(0)$  is 1, both  $(1 - \exp(c*t))$  and  $(1 - \exp(c))$  become zero. We cannot sensibly evaluate 0 divided by 0 (Although mathematically, it is equivalent to the straight line function). We have to take special action for small values of  $c$ , and use the equation  $v = t$  directly.

## Output File Parameters

Output file parameters are controlled by the Output component. The values you choose for the parameters will affect the quality of the sound produced. They will also affect the size of the output file and the amount of time it takes to generate to sound file.

Also bear in mind that a higher quality file containing more data will place a greater load on the system while it is actually being played.

### ***Samples per Second***

The sound file simulates a real sound signal (which is continuous) by storing a sequence of discrete samples. By default, Quack stores 22050 samples per second. This allows almost any audible frequency to be reproduced, although very high pitch components of the sound may be distorted. This does not normally affect the sound quality too badly, because high pitch components usually make up a small part of most sounds.

For very demanding applications, double the sample rate to 44100. This will also double the file size, and double the amount of time taken to produce the file. You could use even higher sample rates, but at very high rates your computer/sound card may not be able to handle the data rate.

If you are experimenting, you might want to select a lower sample rate (11025, or even lower) so that the sound file is generated a bit faster.

### ***Bits per Sample***

Normally, the sound samples are stored as 16 bit integers, which means that there are 65536 different sound levels available.

You can select 8 bits per sample, which only allows 256 different sound levels. This means that the sound is not reproduced as accurately, and some distortion is introduced, but the sound file is halved in size. This is not recommended if you want to achieve high quality sound.

Quack will take approximately the same amount of time to generate a sound file whether it is 8 or 16 bits per sample.

### ***Volume***

It is important to set the correct volume for the output file. If the volume is too high, the signal will be clipped and distorted. If the volume is too low, the signal will not use the whole available range of your soundcard, and so will not be as good as it could be (the signal might also be too

quiet).

Ideally, you should choose a volume so that the peak signal is close to (but not greater than) 1. This can often be achieved simply by setting the volume of all the components in the sound to 1 (which is the default).

If the volume of your sound is out of range for any reason, you can correct it in the Output Component. A message box will appear after the sound has been generated, warning you that the volume is wrong and suggesting the best value to use.

## **Stereo**

It is possible to generate stereo output from Quack. See the technical note on [Working with Stereo Sound](#).

A stereo file is twice as large as the equivalent mono file. It will take slightly longer to produce the file.

## Stereo Sound

Quack simulates a single sound source, which can only be located at one point in space at any particular time.

When Stereo is enabled (by checking the box in the Output component dialog), the source of the sound may be located anywhere between the left and right channels, by setting the *Pan* value. This is a value between 0 and 1 which indicates the position (0 is fully to the left, 1 is fully to the right, 0.5 is halfway between etc.).

Alternatively, you can connect a signal to the *Pan* input of the Output component. This signal may be used to move the position of the sound dynamically (for example, you could use a Sweep signal to move the source from left to right or vice versa).

If you wish to produce stereo sound featuring multiple sound sources, which occupy different positions at different times, you must use Quack to produce several different WAV files. Then use a WAV file editor to mix the files as you require.

Remember that a stereo sound file is twice as large as a mono sound file - it is best to leave stereo disabled if you do not really need it.



## WAV Sample

Reads a WAV sample file in. This allows real sounds to be processed and incorporated into a patch map.

Note that there are (currently) limitations with this function. It only operates single channel (not stereo), uncompressed WAV files.

It is not necessary for the imported file to have the same sample rate as the output file. The imported file will be converted to match the output file.

## Parameters

The dialog box has the following parameters:

- o *Filename* - This is the name of the WAV file you wish to use for the sample.
- o *Speed* - Set this to 1 to play the file at its normal, intended speed. Vary it to replay the file at a different speed (2 play the file at twice normal speed, 0.5 plays it at half speed, etc). The effect is like running a tape recorder at a different speed.

Use can use the **Browse** button to help locate a sound file. After choosing a file, you can use the **Info** button to find out the duration, sample rate and bits per sample of the file. The **Info** button is also a very good way of checking that the sound file is of a type which Quack supports.

## Inputs

If a signal is provided on the Speed input, it controls the speed at which the sound is read. In that case, *speed* is ignored.

## Uses and Examples

The ability to incorporate real sounds into patch maps can give you an excellent “head start” in generating a new sound. The imported sound can be embellished in subtle ways, for example by adding tremolo. It can also be distorted beyond recognition to produce an interesting new sound.

The speed control can be connected to any source, and works well if fed by a low frequency oscillator (with an offset of 1.0 and a volume of, say, 0.05 as a starting point). This can be used to provide various vibrato effects.



## Constant Value

Provides a constant, unchanging signal. This component is also used to provide Tweak Windows, small floating windows which allow key parameters to be varied more easily.

### Parameters

The dialog box has the following parameters:

- *Value* - This is the value of the signal.
- *Tweak Window* - this enables or disables the associated Tweak Window
- *Tweak Caption* - the title which appears in the floating Tweak Window - this is for your own use, so generally just choose a short (8 character max) meaningful name.

### Inputs

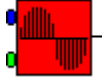
This component has no inputs.

### Uses and Examples

The main use of this component is for the Tweak window it provides.

There are circumstances where it can be useful to have an explicit constant component which can be attached to the inputs of other components. For example, you may wish to add a constant value to a signal using the Merge component.

You should note, though, that in most cases constant values can be defined directly within component edit dialogs. Using the Constant component is (very slightly) less efficient, but sometime clearer, it is your choice.



## Fourier Wave

This component allows you to create your own waveform by mixing harmonics of different amplitudes and phases. You can specify up to 255 harmonics, as well as the frequency and volume of the final waveform.

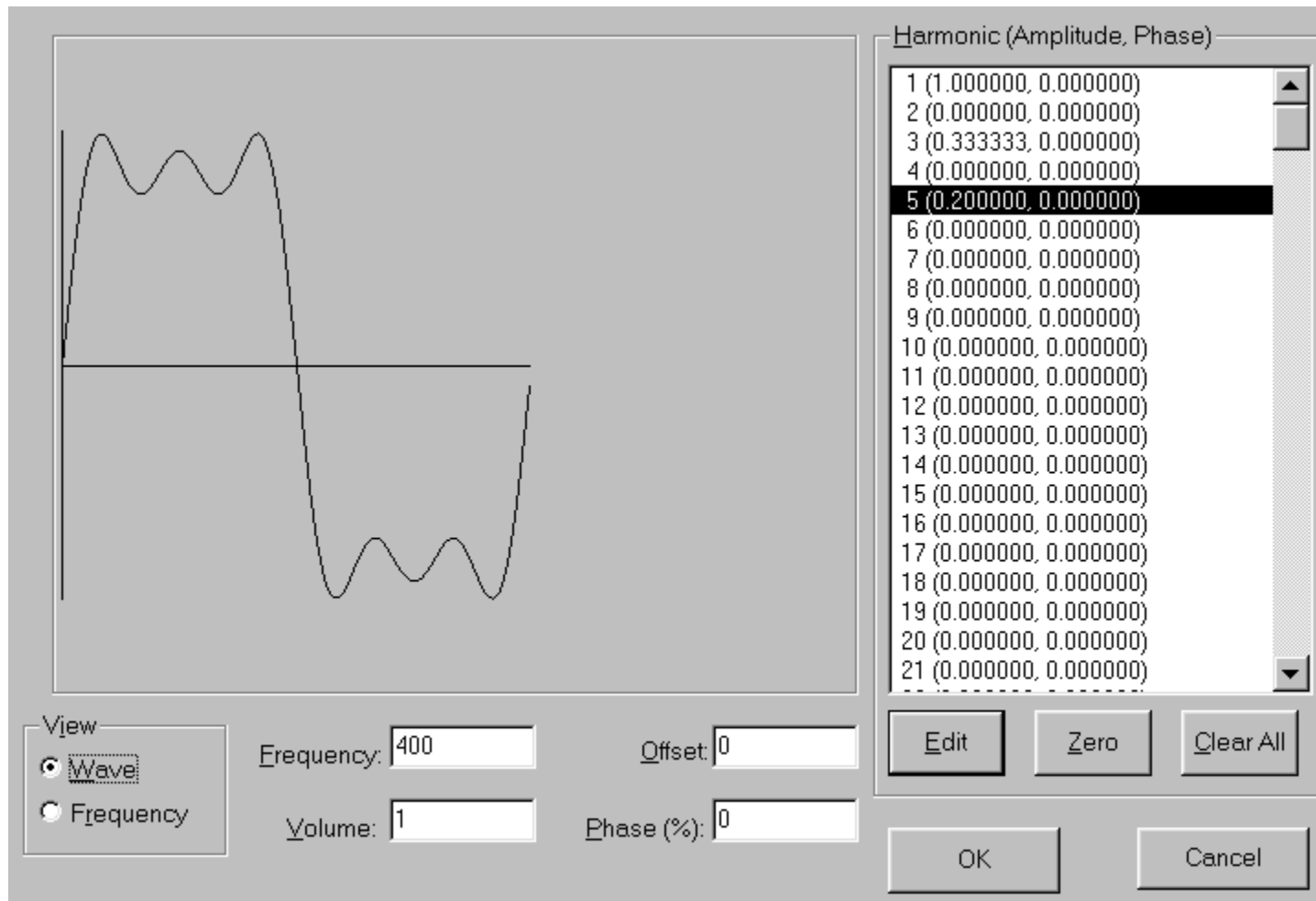
Note - this section assumes that you already have a basic understanding of the principles of Fourier synthesis.

### Parameters

The following parameters may be set in the Fourier Wave Generator dialog box:

- *Frequency* - the frequency in Hertz (or cycles/per second) of the tone generated.
- *Volume* - the peak volume of the signal generated.
- *Offset* - the offset of the average signal, see [Note on waveform offsets](#)
- *Phase* - the initial phase of the waveform.

The power and peak amplitude of the signal is determined by the sum of the components. *Volume* acts as a multiplier on top of this.



The overall *Phase* may be altered so that the wave begins at a different point in its cycle. This takes a value of 0% to 100%, which advances the wave in time by a fraction of a complete cycle.

The *Harmonic* box displays a list of all the harmonics from 1 to 255, showing the amplitude and phase.

To change the values of a harmonic, select it in the list box and choose **Edit**. Enter new values for the *Amplitude* and *Phase* in the dialog box which appears. Note that this allows the individual phase of each component to be modified, but in addition the phase of the entire wave can be changed as described above.

To remove a harmonic, select it in the list box and choose **Zero**. To remove all the harmonics at once, choose **Clear All**.

The box on the left displays a graph of the Fourier waveform. Choose *Wave* in the **View** box to see an actual waveform (time domain). Choose *Frequency* to see a frequency spectrum of the signal (frequency domain). The graph is not to any particular scale.

## Inputs

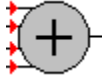
The Fourier Wave Generator has inputs for *Frequency* and *Volume*. These may be connected to the outputs of other components. If they are connected, they totally override the values set in the dialog box.

Note that there is no input to control the *Offset*, see [Note on waveform offsets](#)

## Uses and Examples

Use this component to generate complex tones in the audible range. Choose *Frequency* of anywhere from 20 to 20,000. Set the *Volume* to about 1, and use an offset of 0.0

To generate tremolo or vibrato effects, use a low frequency (less than about 20), and use the *Offset* to create a suitable waveform (see [Note on waveform offsets](#)).



## Mixer

A Mixer component mixes up to 4 input signals to produce a single output signals. The volume of each input signal can be set individually.

### Parameters

The dialog box has 4 parameters:

- o Volume A (B/C/D) - controls the volume of signal A (B/C/D).

### Inputs

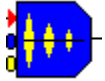
The Merger has inputs for *Signal A*, *Signal B*, *Signal C*, and *Signal D*. These may be connected to the outputs of other components.

The *Signal* inputs are marked by letters A to D.

If any input is not connected, its signal will be set to zero.

### Uses and Examples

This component can be used whenever 2 or more signals need to be mixed in fixed proportions.



## Echo

This component adds single or multiple echoes to a signal.

### Parameters

The dialog box has the following parameters:

- o Delay - the delay between the sound and the first echo
- o Strength - the volume of the the first echo, relative to the original signal.
- o Multiple - checked if multiple echoes are required (ie, an echo of the echo)
- o Max Delay - set this to a little more than the maximum delay you want to use

Note that Max Delay specifies the maximum delay which will be allowed - this is used to decide how much memory to use to store previous signals. Normally, leave this set to a sensible value (a few seconds). If you need to use a long delay, increase the Max Delay accordingly.

### Inputs

There are inputs for the signal, the Delay and the Strength. If Delay and Strength are connected, they completely override the parameters in the dialog.

### Uses and Examples

For a normal echo, use a Strength somewhere between 0 and 1. A Delay of anywhere from 0.2 seconds upwards will give a distinct echo, especially if the sound itself is fairly percussive. Shorter delays will tend to “fill out” the sound without giving a distinct echo.

Remember that, if the echo overlaps the sound, the overall volume is likely to be greater than the original sound.

You can use Strength greater than 1, which will make the echo louder than the original signal, for an unnatural effect. Beware if multiple echoes are enabled, because each echo will be louder than the last, and will eventually cause distortion.

The Strength and Delay can both be varied in time, by connecting other signals. If you vary the delay, you might notice a strange effect on the echo - it will have a slightly different frequency from the original. This is very similar to the Doppler effect when a real signal echoes off a moving target (eg a Police speed radar).



## Fuzz

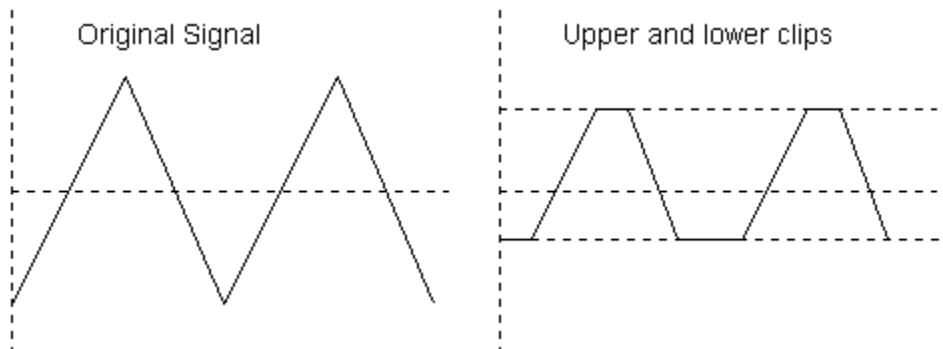
This component introduces deliberate distortion by clipping a signal at a fixed level. There are various types of clipping which can be applied for various effects.

### Parameters

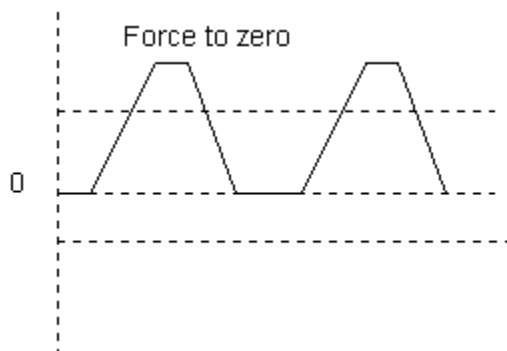
The dialog box has the following parameters:

- ☐ Lower Level Cutoff
- ☐ Upper Level Cutoff
- ☐ Force to zero (on/off)
- ☐ Symmetrical clip (on/off)

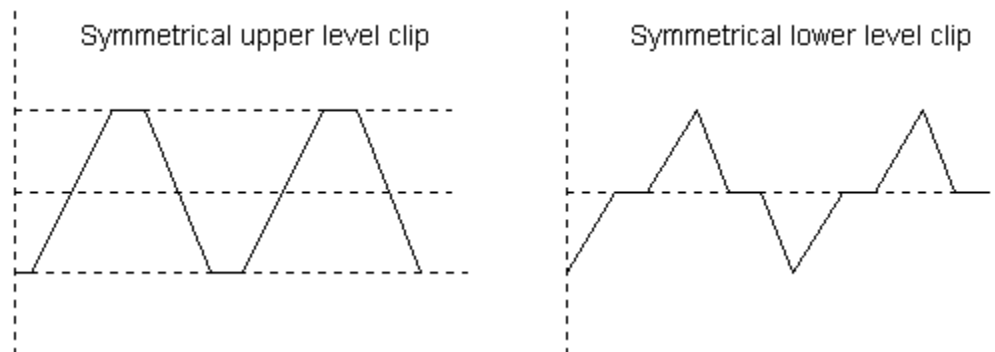
There are also “Clip” check boxes which enable or disable the upper and lower level cutoffs. The diagrams below indicate the effect of the cutoffs when *Force to zero* and *Symmetrical clip* are both disabled:



*Force to zero* has the effect of shifting the signal so that the lower level clipped signal is at zero level (mathematically, it subtracts the Lower Level Cutoff value from the clipped signal).



Symmetrical clipping applies an equivalent clip to both positive and negative going signals. Clip levels must be positive for symmetrical clips. In the diagram below, the lower level clipping is used in conjunction with *Force to zero*.



## Inputs

There is a signal input, and lower/upper level inputs. If connected, the level inputs completely override the values set in the dialog.

## Uses and Examples

This component is useful for introducing distortion to a signal, for effect.

By varying the level (for example, by sweeping it), it is possible to make the distortion vary with time. For a given signal, making the level lower will generally increase the amount of distortion - but it will also decrease the signal volume.

An alternative way to vary the distortion is to keep the level fixed, but to increase the volume of the input signal. This increases the level of distortion while keeping the signal volume more or less constant.



## Music Modulator

This component is used to set oscillator frequencies and sweep ranges in terms of musical notes rather than frequencies. It can also convert a sweep signal to make it musically linear (see below).

### Parameters

The dialog box has the following parameters:

- o Note
- o Octave

This sets the base frequency for the output, according to the musical note selected. For example, if note C octave 5 is selected, the output will be approx 440Hz.

### Inputs

There is one input, Tone. This controls the output. Every unit change in the Tone signal causes a single semitone change in the output, based on whatever note is selected in the dialog. Eg if Tone is 1, the output is one semitone above the base note, if it is -12 the output is one octave below the base frequency.

### Uses and Examples

This component may be used as a convenience when setting oscillator frequencies for musical purposes. Simply connect the output of the music modulator into the frequency input of any oscillator, and you can set the required note directly, without needing to calculate its frequency.

The other use of this component is to make tone transitions musically linear. For example, suppose you want a tone to sweep from C5 up to C7 (ie 440 to 1760 Hz). Just having a sweep go from 440 to 1760 will not sound linear, because the ears hear frequencies on a logarithmic scale (that is, 880 Hz *sounds* as if it is halfway between 440 and 1760, but in a linear sweep the frequency will hit 880 about a third of the way through the signal).

The solution is to generate a linear sweep of 0 to 24 (2 octaves) and feed that through a music modulator. This will generate a musically linear signal.



## **Moving Average (Top hat) Filter**

A moving average filter is a simple low pass filter.

The output of the filter at any point in time is equal to the average of the previous 'n' input samples. This has the effect of smoothing out any rapidly varying signals, hence the lowpass effect.

You can vary the strength of the filter - this operates by mixing a proportion of unfiltered signal in with the filtered signal, decreasing the effect of the filter.

You can also use the filter in high pass mode. In this mode, a proportion of the filtered signal is subtracted from the original signal. By subtracting the low frequencies from the signal, we effectively emphasise the high frequencies.

### **Parameters**

The dialog box has the following parameters:

- o Length - this determines how many samples are used in the average.
- o Strength - this determines how strong the filter effect is. Strength should normally be between 1 (full strength) and 0 (which turns the filter off altogether). Other values are allowed, but may give strange effects.
- o Type - Lowpass or Highpass, see above.

### **Inputs**

The component has a signal input, and an input for the strength. If connected, the strength input completely overrides the parameter in the dialog box.

### **Uses and Examples**

Use this wherever a simple lowpass filter is required.

For example, by feeding noise through a wide moving average filter, you can achieve a signal which varies slowly but randomly with time - this can be used, for example, to modulate the frequency and give a random vibrato effect.

# Registration

QUACK Sound Effects Studio is *shareware*. This means that you are free to use it for evaluation purposes for up to 30 days after you first install it. If you wish to continue using it after this period, you must register.

The purpose of shareware is to allow you to evaluate the product before paying for it. In order for the development of QUACK to continue, we rely on the honesty of our users. To encourage registration, the *unregistered* version has various deliberate limitations which are removed when you register.

You are permitted (and encouraged) to distribute the **unregistered** version of this program freely to others, via any means such as the Internet or CD-ROM provided it is complete and unmodified, including all help files and copyright notices, and that you do not claim to own it, and that you do not charge for it other than a reasonable fee to cover distribution costs. You do not need our permission, but please drop us an email so that we can add you to our mailing list.

You are **not** allowed to distribute the personal password you will receive as a result of registering.

## What you get when you register...

When you register, you receive the right to continue using the copy of QUACK which you already have, under the terms of the Licence.

You will also receive a password which restores the full functionality of the software (removing the shareware limitations stated above). This password is confidential between yourself and Morello, and in registering you agree to take reasonable steps to keep the password confidential.

Note that we do not supply you with a disk copy of the program. This is available for an extra charge, although most users prefer to download the latest version from our website. We recommend that you store 2 backup copies (at least) of the version you have registered, and of course keep your password somewhere safe.

## Payment

Anywhere in the world: 25 UK Pounds, payable by cheque, Postal Order or Money Order.

US Only: 35 US Dollars, by cheque ONLY

Non-US: if you find difficulty in paying in UK Pounds, please contact us for the price in your local currency.

Please **DO NOT** make payments in US Dollars from non-US banks, we cannot accept them.

In addition to your payment, please also tell us:

- Your full name and address (needed for registration)
- Your email address (if you wish to receive your password as quickly as possible)
- The version of QUACK you wish to register.

# Licence

## Shareware

The shareware version is supplied with absolutely no warranty, and is used entirely at your own risk. You may use it for evaluation purposes only, free of charge for a maximum of 30 days. You may also distribute it freely to others under the conditions described under Registration.

## Registered Licence

Upon our acceptance of your registration fee, the following terms apply:

You will receive a password which removes the specific shareware restrictions stated under Registration.

You accept responsibility to make all reasonable efforts to ensure that the password does not become known to others.

You are permitted to use the licenced copy on a single machine or network, provided only one person can use the program at any time.

The program is supplied without warranty.

We do not guarantee the continuous availability of this software via the Internet, or any other source. Please ensure that your shareware copy is adequately backed up before you register.

In any event our liability is limited to the fee you paid for the software.

