# Hive Wavetables

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>2</td>
</tr>
<tr>
<td>Concepts</td>
<td>3</td>
</tr>
<tr>
<td>Some Terminology</td>
<td>3</td>
</tr>
<tr>
<td>Commands</td>
<td>4</td>
</tr>
<tr>
<td>Info</td>
<td>4</td>
</tr>
<tr>
<td>NumFrames</td>
<td>4</td>
</tr>
<tr>
<td>Seed</td>
<td>4</td>
</tr>
<tr>
<td>Wave</td>
<td>5</td>
</tr>
<tr>
<td>Spectrum</td>
<td>5</td>
</tr>
<tr>
<td>Phase</td>
<td>5</td>
</tr>
<tr>
<td>Import</td>
<td>6</td>
</tr>
<tr>
<td>Export</td>
<td>6</td>
</tr>
<tr>
<td>Move</td>
<td>6</td>
</tr>
<tr>
<td>Interpolate</td>
<td>7</td>
</tr>
<tr>
<td>Normalize</td>
<td>7</td>
</tr>
<tr>
<td>Envelope</td>
<td>7</td>
</tr>
<tr>
<td>Operators</td>
<td>8</td>
</tr>
<tr>
<td>Mathematical Functions</td>
<td>9</td>
</tr>
<tr>
<td>Complex / DSP functions</td>
<td>9</td>
</tr>
<tr>
<td>Sample Accessor Functions</td>
<td>10</td>
</tr>
<tr>
<td>Variables</td>
<td>10</td>
</tr>
<tr>
<td>Waveform Blend Modes</td>
<td>10</td>
</tr>
</tbody>
</table>
Hive can load wavetable files in either .wav or .uhm format.

The former is easily described: Mono or stereo WAV files are interpreted by default as having a cycle length of 2048 samples per frame – the number of frames is calculated from the total size of the file and is limited to 256. To specify other samples-per-frame values, the file name must end with “-WT” followed by the number e.g. MyWavetable-WT512.wav (or 64, 128, 256, 1024).

Enough about WAV already. Let's get straight to the interesting stuff… UHM.

Hive’s wavetable scripts are readable text files with the extension .uhm. These scripts create wavetables step-by-step by interpreting a list of commands / formulas. Here are two examples just to set the stage (it doesn’t really matter how little or how much you understand at first):

```plaintext
 NumFrames = 5 // number of waveforms (“frames”) in the wavetable

 // First, a sawtooth in frame 0
 // Learn: “phase” goes from 0 to 1
 // Learn: to work on specific frames, specify Start and End
 Wave start=0 end=0 “-phase”

 // Then a sine wave in frame 1
 Wave start=1 end=1 “sin(2*pi*phase)”

 // Next, a triangle in frame 2
 Wave start=2 end=2 “-abs(1-2*phase)”

 // A square in frame 3
 // Learn: comparisons (’<’ here) return 1 if true, 0 if false
 Wave start=3 end=3 “phase<0.5”

 // Narrower pulse in frame 4
 Wave start=4 end=4 “phase<0.05”

 // Learn: collect useful tricks...
 Spectrum lowest=0 highest=0 “0” // DC-removal trick
 Normalize start=0 end=3
 Normalize start=4 end=4 db=-5 // different level for the narrow pulse
```
This short example creates 256 frames:

```plaintext
// Emulate a synced square wave
NumFrames = 256
Wave "-1 + 2 * (frac(phase*(7*table+1)) > 0.5)"
```

**Concepts**

A method of defining wavetables using script files containing lists of commands. The main tool is a parser which interprets mathematical expressions and variables.

Basic variables are the frame index within the wavetable and the current sample value within the frame (see Some Terminology below).

More elaborate functions can access any wavetable, or create filters and distortion. Synthesis methods such as PD (phase distortion), FM or fractal resonance / sync can be achieved with just a few lines of code.

- Scripted wavetables load just like .wav files. They reside in the same folders and are stored in the presets in the same way (i.e. by reference only)
- Scripted wavetables can load frames from other wavetables and recombine them, even layer or morph between them
- Scripts can be used to edit any existing wavetables, e.g. normalise or filter them
- In addition to the resulting wavetable, a script can store and retrieve intermediate calculations from two (temporary) auxiliary buffers. Like the memory functions (M+, MR) in a calculator.
- Most operations can be performed on a defined range of frames, even backwards if End < Start
- Most operations can use blending modes (e.g. add, subtract, multiply…) to impose their result on the existing wavetable buffer. Similar to Photoshop layer blending modes.
- The arguments of most commands default to useful values i.e. they don’t all have to be stated explicitly in each and every line of script
- Like wavetables from .wav files, Hive scripts support between 1 and 256 frames, with each frame representing a fixed-length waveform (2048 samples).

**Some Terminology**

To avoid any confusion…

- **wavetable** Our “wavetable” is the set of one or more waveforms.
- **frame** A frame is the position that a waveform occupies within a wavetable.
- **waveform** A set of samples (here: 2048) which a wavetable oscillator cycles through
- **scan** Cause an oscillator to select (or interpolate between) frames within a wavetable
- **phase** The cycle of a waveform from 0.0 to 1.0 (same as position 0.0 of the following cycle)
- **index** The position of a sample within the waveform (0-2047). Equal to Phase * 2048
- **buffer** The sample memory a wavetable occupies (main, aux1 or aux2)
Commands

Info 
Display a short message in the information field

NumFrames
Sets the number of frames in the wavetable

Seed
Sets random generator seed

Wave
Parses formula on time domain Wavetable

Spectrum
Parses formula on magnitudes of frequency domain Wavetable

Phase
Parses formula on phase info of frequency domain Wavetable

Import
Loads a .wav or .uhm file into specific frames

Export
Saves a buffer as a .wav file

Move
Copies frames

Interpolate
Interpolates between a range of frames, e.g. using formatize

Normalize
Adjusts the volume of the frame to a target level (dB)

Info

// This message will appear in the wavetable info field
Info "A collection of standard waveforms for subtractive synthesis"

NumFrames

Sets the number of frames in the wavetable. Defaults to 256.

Example:
NumFrames = 201

Seed

This command assures variation between random values in multiple scripts. Each script retains the same sequence of values so that the oscillator will sound the same every time it is loaded.

Example:
Seed = 206452865

Seed is used by all three random functions:

rand 
A random value for each operation

randf
A random value for the current frame (remains the same for all samples)

rands
A random value for the current sample (remains the same for all frames)
Wave

Runs the parser on the wavetable. The following options are available:

- **Start=N**, where N is the first frame to process. Defaults to 0.
- **End=N**, where N is the last frame to process. Defaults to NumFrames.
- **Blend=X**, where X is a waveform blend mode (replace, add, multiply...). Defaults to replace.
- **Direction=X**, where X is forward or backward. Direction samples are processed. Defaults to forward.
- **“Formula in quotes”** (required)
- **Target=X**, where X is main, aux1 or aux2, defining the target buffer

Example:
```
// Blends 40 of the frames in the current buffer
Wave Start=10 End=50 Blend=Add Direction=backward "sin(2*pi*phase-x)"
```

Spectrum

Runs the parser on the spectrum (magnitudes) of the wavetable. The following options are available:

- **Start=N**, where N is the first frame to process. Defaults to 0.
- **End=N**, where N is the last frame to process. Defaults to NumFrames.
- **Lowest=N**, where N is the lowest partial to process. Defaults to 1, but can also be set to 0 (DC)
- **Highest=N**, where N is the highest partial to process. Defaults to 1024
- **Blend=X**, where X is a waveform blend mode (replace, add, multiply...). Defaults to Replace.
- **Direction=X**, where X is forward or backward. Direction samples are processed. Defaults to forward.
- **“Formula in quotes”** (required)
- **Target=X**, where X is main, aux1 or aux2, defining the target buffer

Example:
```
Spectrum Start=2 End=155 Direction=backward "x/(1+1024*phase)"
```

Phase

Runs the parser on the spectrum (phase information) of the wavetable. The following options are available:

- **Start=N**, where N is the first frame to process. Defaults to 0.
- **End=N**, where N is the last frame to process. Defaults to NumFrames.
- **Lowest=N**, where N is the lowest partial to process*. Defaults to 1 (fundamental). 1 is also the lowest possible, as DC has no phase information.
- **Highest=N**, where N is the highest partial to process*. Defaults 1023 (highest partial with phase info).
- **“Formula in quotes”** (required)
- **Target=X**, where X is main, aux1 or aux2, defining the target buffer

Example:
```
// set all phases to 0 for frames 0-255 in aux1 buffer
Phase Start=0 End=255 "0" Target=aux1
```
Import

Loads a .wav or .uhm file and writes/blends it (or a portion thereof) into the target wavetable. The source file must be in the same directory as the script. The following options are available:

- **Start=N**, where N is the first frame to be extracted from the source file. Defaults to 0.
- **End=N**, where N is the last frame to be extracted from the source file. Defaults to NumFrames.
- **Blend=X**, where X is a waveform blend mode (replace, add, multiply...). Defaults to Replace.
- **From=N**, where N is the first frame in the target buffer which the extracted frames are written to. Defaults to 0.
- **Target=X**, where X is main, aux1 or aux2, defining the target buffer

Example:

/* Extract all available frames, or NumFrames-10 (whichever is smaller) and write them into frames 10+ of the main buffer) */

Import Start=0 From=10 "some wavetable.wav"

Export

Saves the main, aux1 or aux2 buffer as a .wav file into the same directory as the script.

- **Source=X**, where X is main, aux1 or aux2, defining the target buffer. The default is main

Example:

/* saves a wavetable with 17 frames (default is 256) as myFile.wav. Info is stored in the .wav file as metadata, and will appear as text in Hive whenever the wavetable is loaded. See the Info command. */

Info “Explanatory text”
NumFrames = 17
Export Source=aux1 "myFile.wav"

Move

Copies frames within a wavetable. The following options are available:

- **Start=N**, where N is the first frame to be processed. Defaults to 0.
- **End=N**, where N is the last frame to be processed. Required.
- **To=N**, where N is the first frame to be overwritten. Required.
- **Blend=X**, where X is a waveform blend mode (replace, add, multiply...). Defaults to Replace.
- **Target=X**, where X is main, aux1 or aux2 buffer

Example:

// Multiply 101 frames of buffer Aux2 with the same frames from Aux1
Move source=aux1 target=aux2 Blend=multiply start=0 end=100
Interpolate

Interpolates between frames within a wavetable. Details of the “morph” types and the last 3 options here will be explained at a later date. The following options are available:

- **Start=N**, where N is the start frame of the interpolation. Defaults to 0.
- **End=N**, where N is the end frame of the interpolation. Defaults to NumFrames.
- **Type=X**, where X is an interpolator type (switch, crossfade, spectrum, zerophase, morph1, morph2). Defaults to crossfade.
- **Target=X**, where X is main, aux1 or aux2 buffer
- **Snippets=X** (1-500) maximum number of fragments to be morphed (Morph1/2 only).
- **Threshold=X** (-120 - 0) threshold for identifying snippets (Morph1/2 only)
- **Weighting=X** where X is none, distance, level or both (Morph1/2 only)

Example:
// Fill frames 1-99 with intermediate states between frames 0 and 100.
Interpolate Start=0 End=100 Type=morph1

Normalize

Normalizes frames within a wavetable. The following options are available:

- **Start=N**, where N is the first frame to be processed. Defaults to 0.
- **End=N**, where N is the last frame to be processed. Defaults to NumFrames.
- **Metric=X**, where X is “RMS”, “Peak”, “Average” or “Ptp” (peak to peak). Defaults to RMS.
- **dB=M**, where M is a value in dB (typically 0dB for Peak). Defaults to 0.00.
- **Base=X**, where X is "All" or “Each” (individual frames are normalized). Defaults to Each.
- **Target=X**, where X is main, aux1 or aux2

Example:
Normalize base=each

Envelope

Creates an envelope with up to 8 segments to be used within the formula parser. Using "env(frame)" within a formula gives us an envelope value based on frames. Using "env(index)" gives us an envelope based on samples. However, you can use any values to scale or shift the envelope from frame to frame, for instance.

- **T1=N, T2=N, ... T8=N**, where N are integer time values, typically mapped to indices
- **L0=N, L1=N, ... L8=N**, where N are floating point levels, typically from 0.0 to 1.0
- **Curve=X**, where x is "linear", "exponential", logarithmic or “quadric”: curvature of each segment

Example1:
// Use an envelope to create a very sweet-sounding “fin” triangle
Envelope curve=exponential L0=0 T1=0.25 L1=1 T2=0.5 L2=-1 T3=0.25 L4=0
Wave “env(phase)”

Example 2:
// Use an envelope as window for a fractalized square
NumFrames=128
Envelope L0=1 T1=2048 L1=0 // simple saw envelope
Wave "env(index)*(1-2*((phase*(1+0.25*frame)%1)>0.5))"
Operators

<table>
<thead>
<tr>
<th>Basic Math</th>
<th>Comparisons</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>addition</td>
</tr>
<tr>
<td>-</td>
<td>subtraction</td>
</tr>
<tr>
<td>*</td>
<td>multiplication</td>
</tr>
<tr>
<td>/</td>
<td>division</td>
</tr>
<tr>
<td>%</td>
<td>Floating point modulo</td>
</tr>
<tr>
<td>^</td>
<td>Exponent, x to the power of y</td>
</tr>
<tr>
<td>&lt;</td>
<td>smaller than</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>smaller than or equal to</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>equal to</td>
</tr>
<tr>
<td>!=</td>
<td>not equal to</td>
</tr>
</tbody>
</table>

Example 1:
// The easiest method of creating a square wave
Wave “1-2*(phase>0.5)”

Example 2:
// The hardest method of creating a triangle wave
// (see Tut02 Classic Waves Part 2.uhm)
Spectrum "1/(index*index)*((index % 2)==1)*(1-2*((index % 4)==3))”
Mathematical Functions

Note that the trigonometric functions expect x in radians. So to create a sine wave with a phase from 0 to 1, we write \( \sin(2\pi\times\text{phase}) \)

<table>
<thead>
<tr>
<th>Trigonometric</th>
<th>Floating Point</th>
<th>Mathematical</th>
</tr>
</thead>
<tbody>
<tr>
<td>acos(x)</td>
<td>arccosine</td>
<td>abs(x)</td>
</tr>
<tr>
<td>asin(x)</td>
<td>arcsine</td>
<td>ceil(x)</td>
</tr>
<tr>
<td>atan(x)</td>
<td>arctangent</td>
<td>floor(x)</td>
</tr>
<tr>
<td>atan2(x,y)</td>
<td>arctangent2</td>
<td>frac(x)</td>
</tr>
<tr>
<td>cos(x)</td>
<td>cosine</td>
<td>round(x)</td>
</tr>
<tr>
<td>cosh(x)</td>
<td>hyperbolic cosine</td>
<td>select(x,y,z)</td>
</tr>
<tr>
<td>sinh(x)</td>
<td>hyperbolic sine</td>
<td></td>
</tr>
<tr>
<td>tan(x)</td>
<td>tangent</td>
<td></td>
</tr>
<tr>
<td>tanh(x)</td>
<td>hyperbolic tangent</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>exp(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>fac(x)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ln(x) or log(x)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>log10(x)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>log base 10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pow(x,y)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sqrt(x)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lin_db(x)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>db_lin(x)</td>
</tr>
</tbody>
</table>

Complex / DSP functions

\[ \text{env}(x) \] where \( x \) is the time position/index of the 8-segment envelope (see Envelope above). The result is the envelope level at that time position/index

\[ \text{lowpass}(x, \text{cutoff}, \text{resonance}) \] where \( x \) is the sample and \( \text{cutoff} / \text{reso} \) are values between 0 and 1

\[ \text{bandpass}(x, \text{cutoff}, \text{resonance}) \] where \( x \) is the sample and \( \text{cutoff} / \text{reso} \) are values between 0 and 1

\[ \text{highpass}(x, \text{cutoff}, \text{resonance}) \] where \( x \) is the sample and \( \text{cutoff} / \text{reso} \) are values between 0 and 1

Example:

```
// create a 256-frame lowpass filtered sawtooth
Wave "lowpass(2*phase-1, table+0.1, 0.5)"
```
Sample Accessor Functions

For access to any samples in any frame in any buffer (more flexible than x, main, aux1, aux2):

- `main_fi(frame, index)` the sample at index (0-2047) of frame (0-255) in the main buffer
- `main_fp(frame, phase)` the sample at phase (0-1) of frame (0-255) in the main buffer
- `aux1_fi` via index
- `aux1_fp` via phase
- `aux2_fi` via index
- `aux2_fp` via phase

Example:

```
// reverse the order of frames in a 101-frame wavetable
Wave target=aux1 "main_fi(100-frame, index)"
```

Variables

- `e`: 2.71828182845904523536
- `pi`: 3.14159265358979323846
- `x`: Current sample/magnitude/phase in wavetable of current target buffer
- `y`: Previous result of the expression parser
- `frame`: Current frame
- `table`: Current frame, normalized to the current process loop (0-1)
- `index`: Current sample index within the frame
- `phase`: Current sample, normalized to one waveform cycle (0-1, index/2048)
- `rand, randf, rands`: See “Seed” above
- `main, aux1, aux2`: Like x/input, but from the respective buffer

Waveform Blend Modes

- `replace`: source replaces target
- `add`: source is added to target
- `sub`: source is subtracted from
- `multiply`: target is multiplied with source
- `multiplyAbs`: (x + fabs(y))
- `divide`: sign(x\(\times\))\(1-fabs(x))\(1-fabs(y))
- `divideAbs`: sign(x)\(1-fabs(x))\(1-fabs(y))
- `min`: minimum, the smaller absolute value
- `max`: maximum, the larger absolute value (good for emulating formants when used with Spectrum)