MIDI: sound control

Juan P Bello

Interpreting MIDI

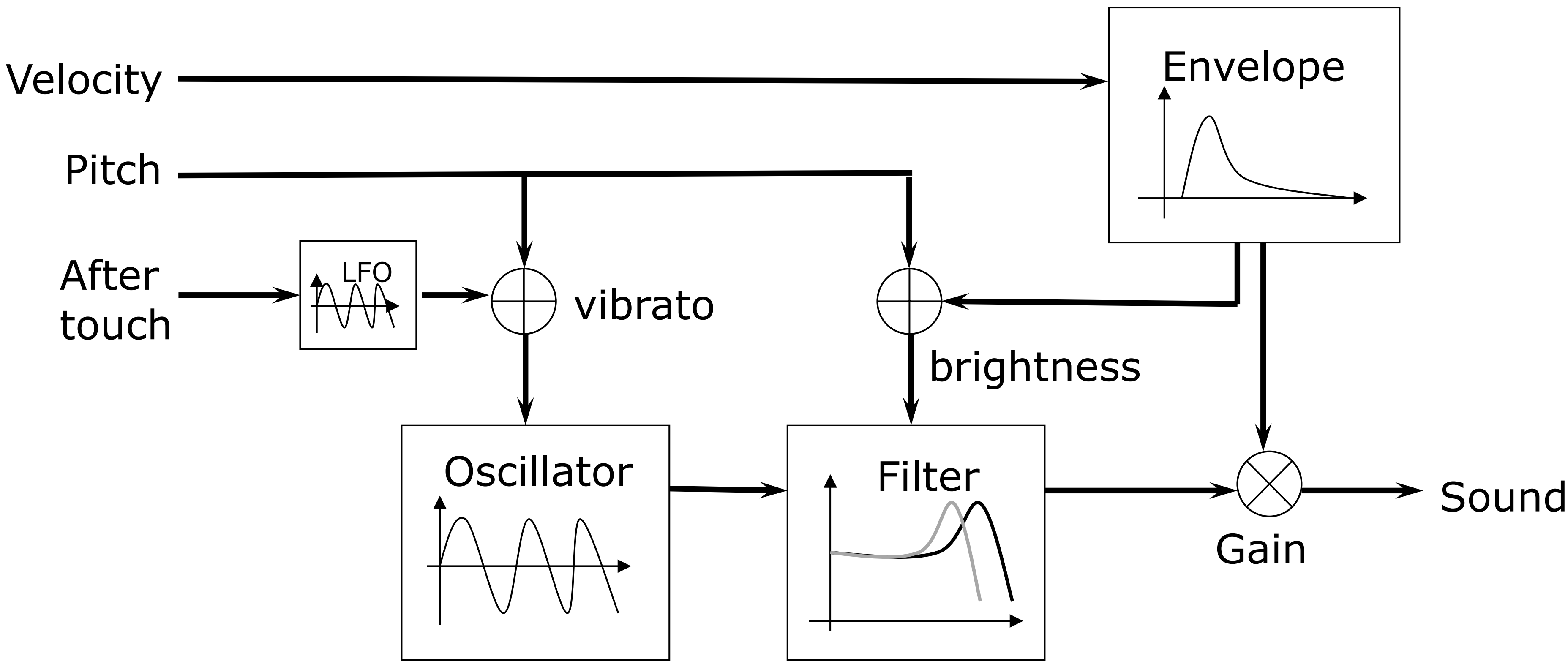
- MIDI code is the language used to communicate between devices
- Still, each device might decide to interpret the code in its own way according to its capabilities and functionalities
- The MIDI implementation for a particular sound generator should be described by a so-called MIDI implementation chart which is commonly part of its manual
- It indicates which messages are received and transmitted, as well as limitations and uncommon features
- For example a device might be able to receive all possible note numbers (0-127) but only able to transmit a sub-set of those (e.g. 16-115)

Polyphony and voices

- Most current sound generators (e.g. synthesizers and samplers) are polyphonic (with a cap on the maximum number of notes)
- When the polyphony of a device is exceeded, a predefined strategy is activated that governs how to handle the extra notes
- Different strategies include: releasing the “oldest” or “quietest” notes first, or simply not accepting any new notes until current notes are released
- The degree of polyphony of a device is different from the number of voices it can generate
- How the polyphony is distributed amongst the voices is, again, device specific
- Allocating polyphony according to demand is perhaps the most common approach nowadays
- Current systems are capable of generating several voices at the same time, independent of polyphony considerations

Velocity and aftertouch

- Note velocity and aftertouch can be used to control specific functions within the sound generation chain



- Note off velocity, uncommon in many devices, can be used to control the release time of the note, reverberation time, etc

General MIDI (1)

- Although program change can be used to select voice patches, there is no guarantee that the same voice will be recalled by the same message in different instruments
- E.g. program change message 10 may refer to “trombone” on one instrument and “ukulele” on another.
- This makes the exchange of songs between instruments very difficult, as their reply will be different on every device.
- General MIDI is an attempt to standardize the behavior of sound generators in the presence of MIDI files.
- Three types: General MIDI Level 1 (GM1), GM Lite (GML) and GM Level 2 (GM2)

General MIDI (2)

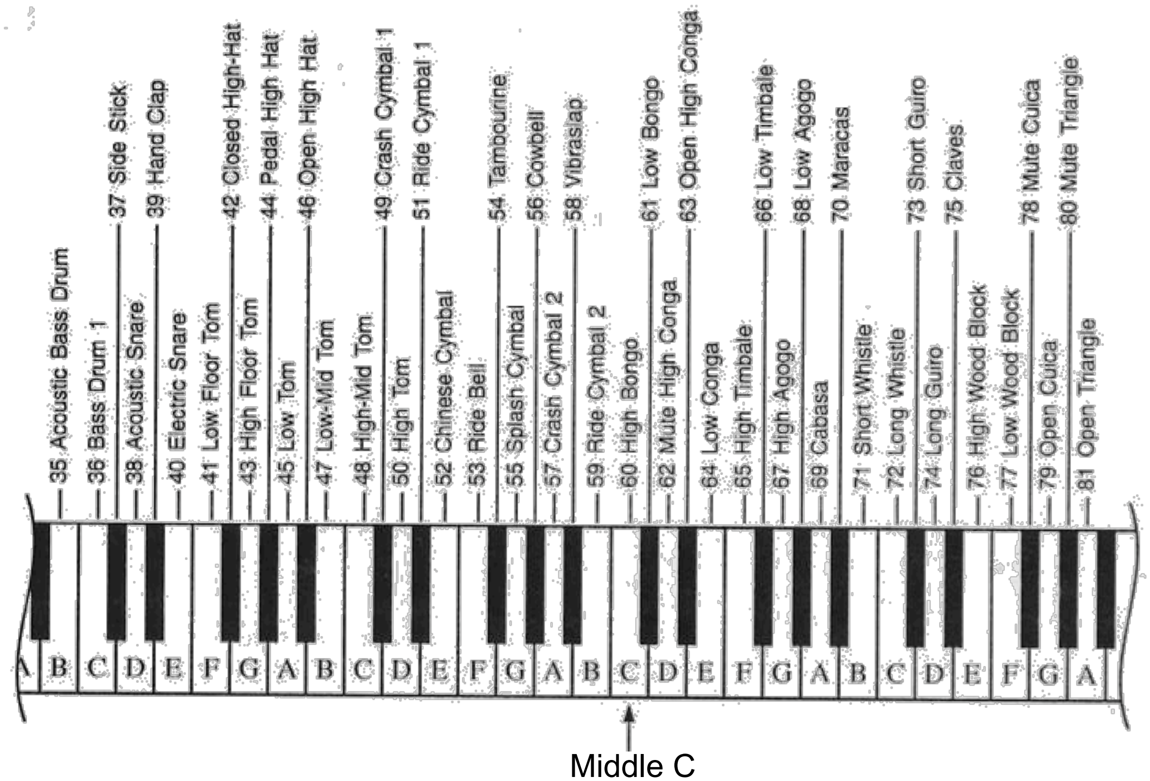
- GM 1 specifies a standard voice map:

Program (decimal)	Sound	Program (decimal)	Sound
0-7	Piano	64-71	Reed
8-15	Chromatic Percussion	72-79	Pipe
16-23	Organ	80-87	Synth lead
24-31	Guitar	88-95	Synth pad
32-39	Bass	96-103	Synth effects
40-47	Strings	104-111	Ethnic
48-55	Ensemble	112-119	Percussive
56-63	Brass	120-127	Sound effects

- Precise voice names can be found in the GM documentation

General MIDI (3)

- The exception is channel 10, where percussion sounds (organized as a drum-kit) are ALWAYS allocated.
- For this channel GM also defines the map between note numbers and “drum-kit” sounds

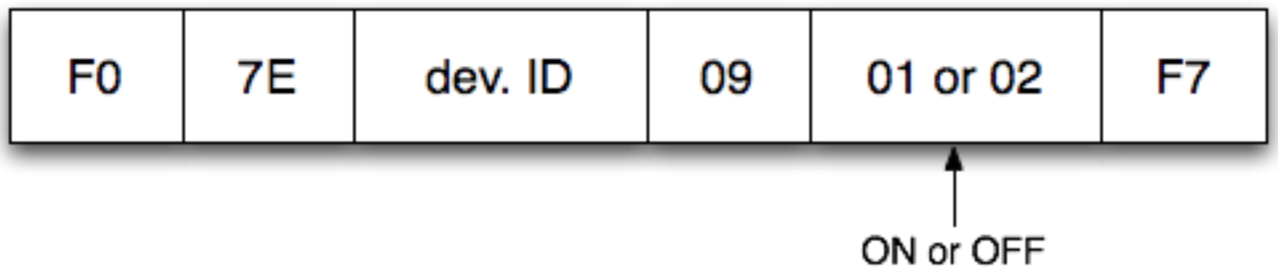


General MIDI (4)

- GM specifies that a compliant sound generator should be able to receive 16 simultaneous and polyphonic channels (with a different voice per channel)
- The minimum required polyphony is 24, being dynamically allocated between the voices
- GM modules should be sensitive to velocity, pitch bend (defaulted to ±2 semitones) and channel aftertouch.
- Required control messages: modulation wheel (01), channel volume (07), pan (0A), expression controller (0B), sustain pedal (40), reset all controllers (79), and all notes off (7B).
- Required RPNs: pitch bend sensitivity (00), fine tuning (01), and coarse tuning (02).

General MIDI (5)

- GM modules can operate in modes different from GM. Two universal SysEx messages are used to switch GM on and off:



- GM Lite is a simplified version of GM 1 for devices with limited processing power (e.g. mobile phones, PDAs)
- GM 1 songs will replay with acceptable quality but some information may be lost.
- GM 2 extends the functionalities of Level 1 by incorporating: selection of voice banks, 32-voice polyphony, MIDI tuning, RPN controllers and a number of universal SysEx messages. Percussion kits can run on channel 11 (as well as 10).
- GM 2 is backward compatible with GM 1.

SPMIDI

- Scalable Polyphonic MIDI (SPMIDI), like GM lite, is a standard for mobile devices with limitations in battery life and processing power

- It is widely used for the control of synthetic sounds in ring tones and multimedia messaging.

- It is different from GM on its flexibility to adapt to the limitations of a device, e.g. by scaling polyphony accordingly

- Furthermore, it allows the MIDI creator to define what should happen when polyphony is limited, instead of using conventional “note stealing”

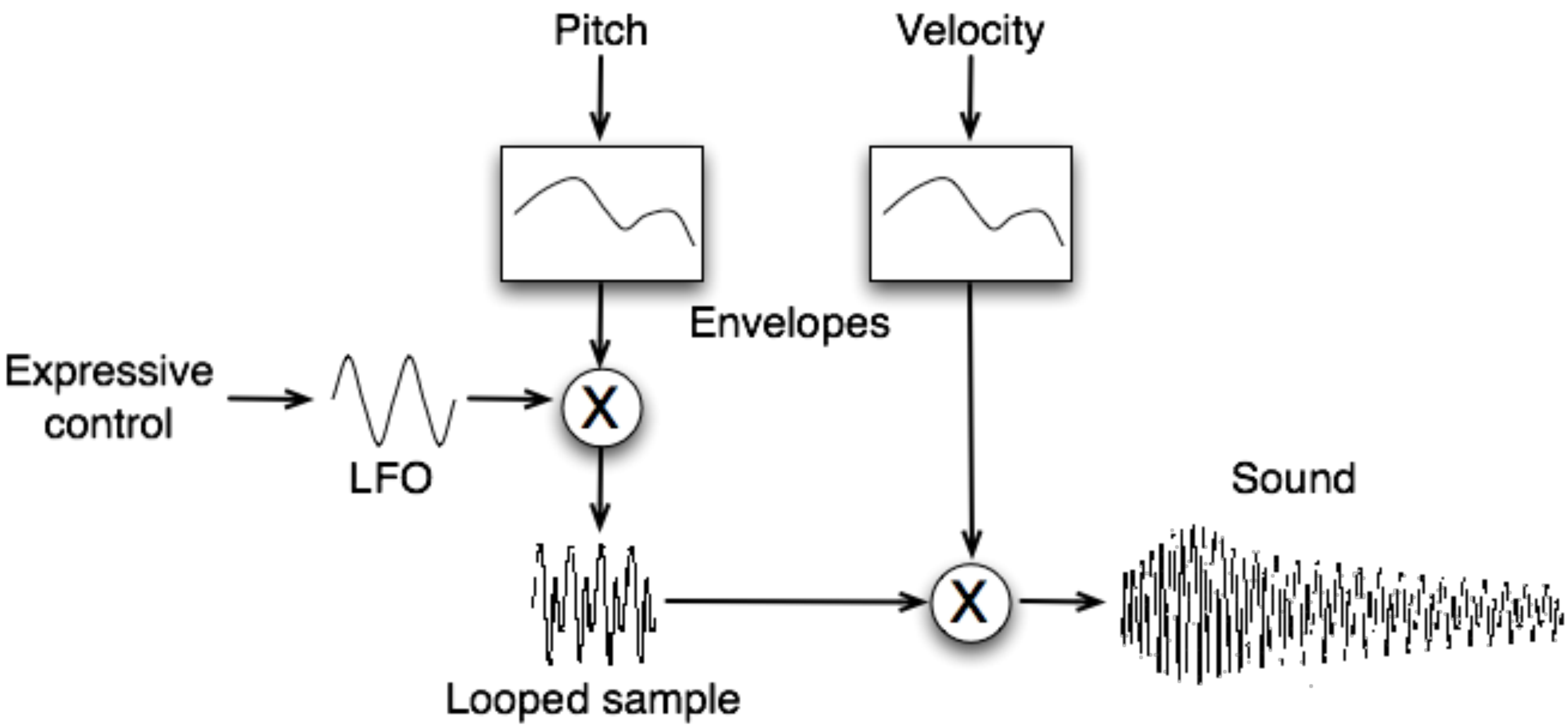
- Channel masking: Through a set-up message at the beginning of the transmission, it allows for certain channels to be defined as higher priority than others.

DLS and Soundfonts

- There are many different methods for the description and exchange of synthetic sounds
- Examples include: downloadable sounds (DLS), sound fonts and Structured Audio (SA) sample bank format
- DLS is an MMA specification that enables synthesizers to be programmed using sounds downloaded from a variety of sources
- This is to allow MIDI not only to control “score” information in a universal manner, but also the exact sounds to be used
- The idea is to make the end result of the playback more predictable across platforms

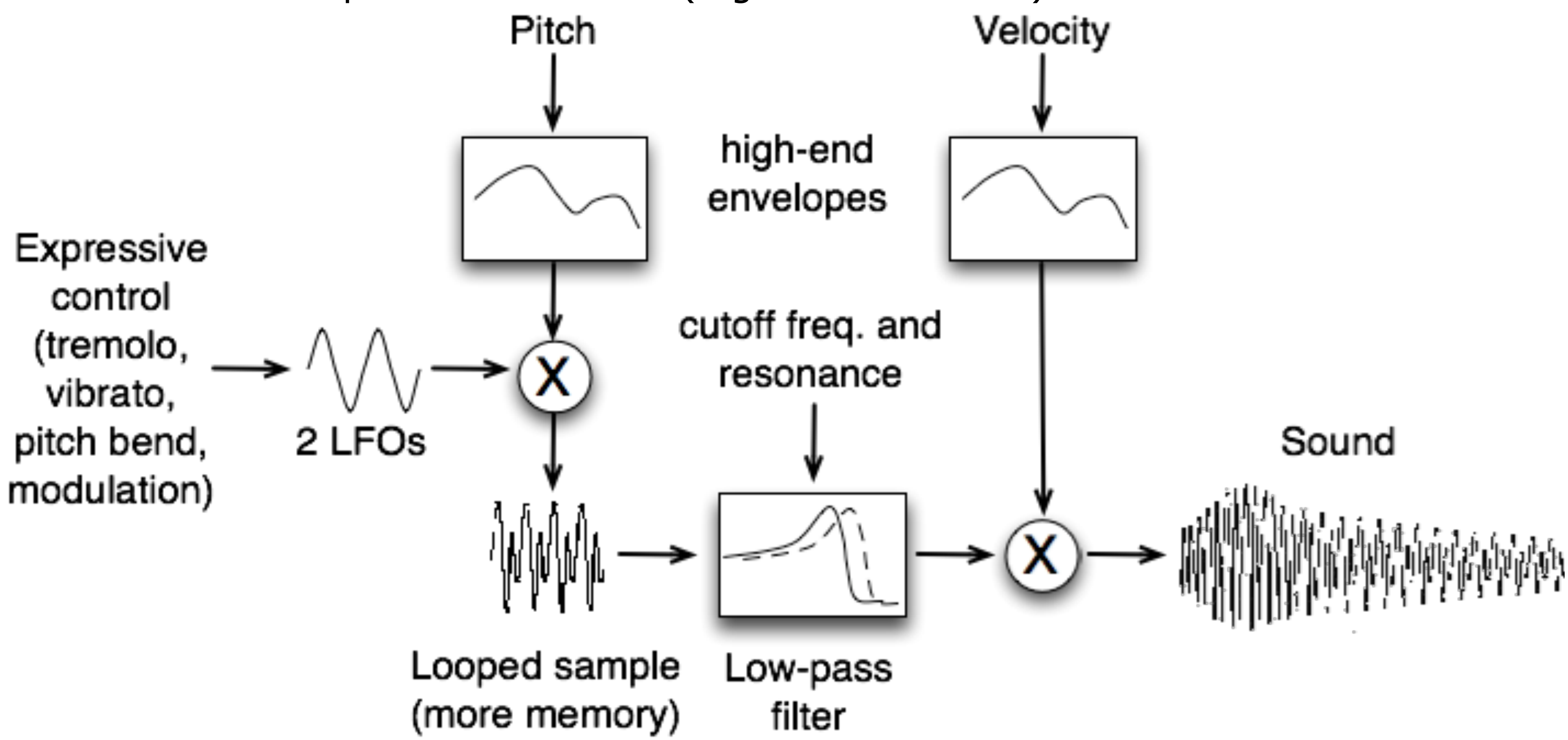
DLS Level 1

- DLS Level 1 (1999) contains a specification for DLS devices and a file format for holding the sound descriptions
- It is based on a simple wavetable synthesizer



DLS Level 2

- DLS Level 2 is more advanced, requiring more memory space.
- It has been adopted as the MPEG-4 Structured Audio sample bank format
- It is not very different from Emu's SoundFonts which have been widely used in computer sound cards (e.g. Sound Blaster)

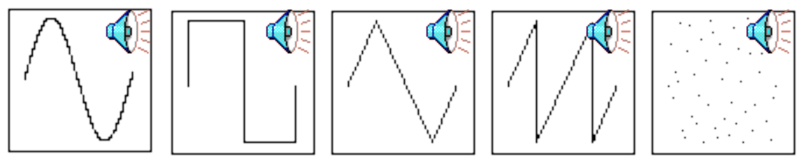


Structured audio

- Structured Audio (SA) is an standard for low-bit representation of digital audio. It is part of the MPEG-4 audio standard
- SA is generative in that it decomposes audio into a collection of instruments (orchestra) and the “score” that is used to control those instruments
- Each function has its own language: structured audio orchestra language (SAOL) and structured audio score language (SASL)
- SAOL, an offspring of the synthesis language CSound, is more general than DLS or similar specifications, since is not restricted to wavetable synthesis.
- It allows sound generation using any digital synthesis method including additive, subtractive, AM, FM, Physical modeling, etc.
- Likewise, SASL is more versatile than MIDI. It allows the use of MIDI data by including previsions for the translation of MIDI commands into SAOL events.

Oscillators

- All synthesis starts with an oscillation used to generate a raw repeating signal/waveform.
- Common oscillator-generated signals include: sine, triangle, sawtooth, square and random noise

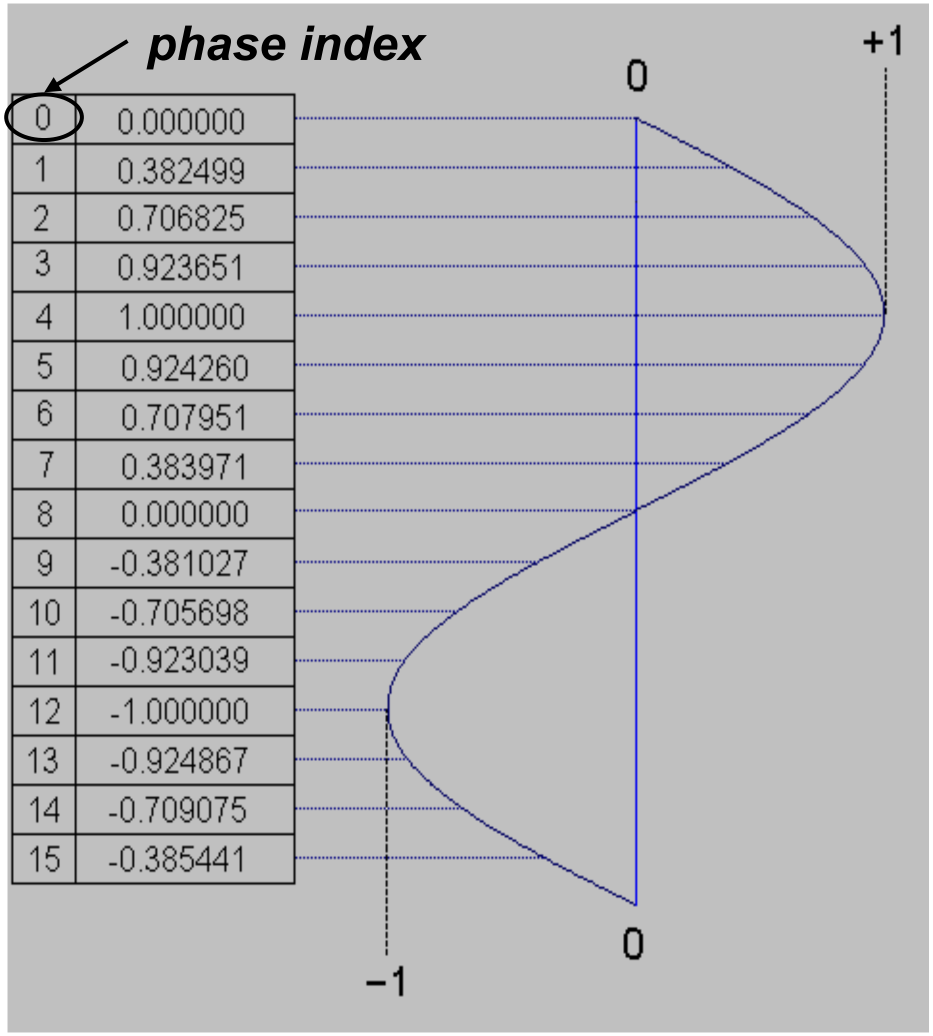


- These signals have different spectral complexity - they are more or less rich harmonic content
- In digital synthesis, the sound signal is generated and/or stored as a stream of numbers (samples)
- These samples can be generated algorithmically every time a synthetic sound is required - not the most efficient solution

Wavetables (1)

- More efficient solution:
  - Exploit the repetitiveness of musical sound
  - Calculate (or record) numbers once
  - Have the numbers stored in a memory list: a <u>wavetable</u>

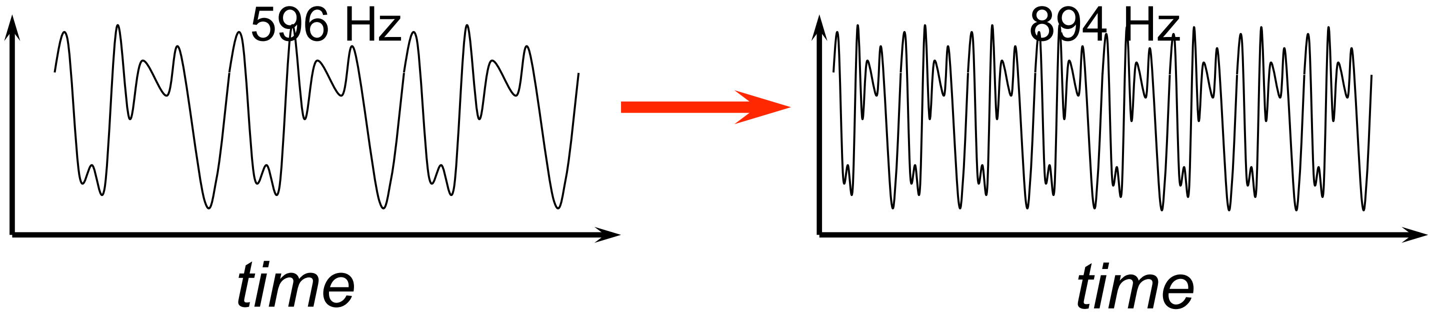
- Called back whenever necessary (repetitive scanning)
- the waveform is unchanged (fixed) during scanning



0	0.000000
1	0.382499
2	0.706825
3	0.923651
4	1.000000
5	0.924260
6	0.707951
7	0.383971
8	0.000000
9	-0.381027
10	-0.705698
11	-0.923039
12	-1.000000
13	-0.924867
14	-0.709075
15	-0.385441

Wavetables (2)

- The frequency of the sound is: cycles * sampling frequency / length of table (e.g. NC = 2, fs = 100kHz, L =1000; *f* = 200 Hz)
- We can modify the scanning rate, i.e. modify the hop size or phase increment at which the table will be read
- Changes on this increment effectively shrink the table, thus changing the frequency of the sound (pitch shift):
  f_0 = (phase_increment * cycle * fs)/L
- Phase increments are integer values, since they refer to memory samples



- However, for most pitch shifts, integer increments (and memory pointers) are not adequate

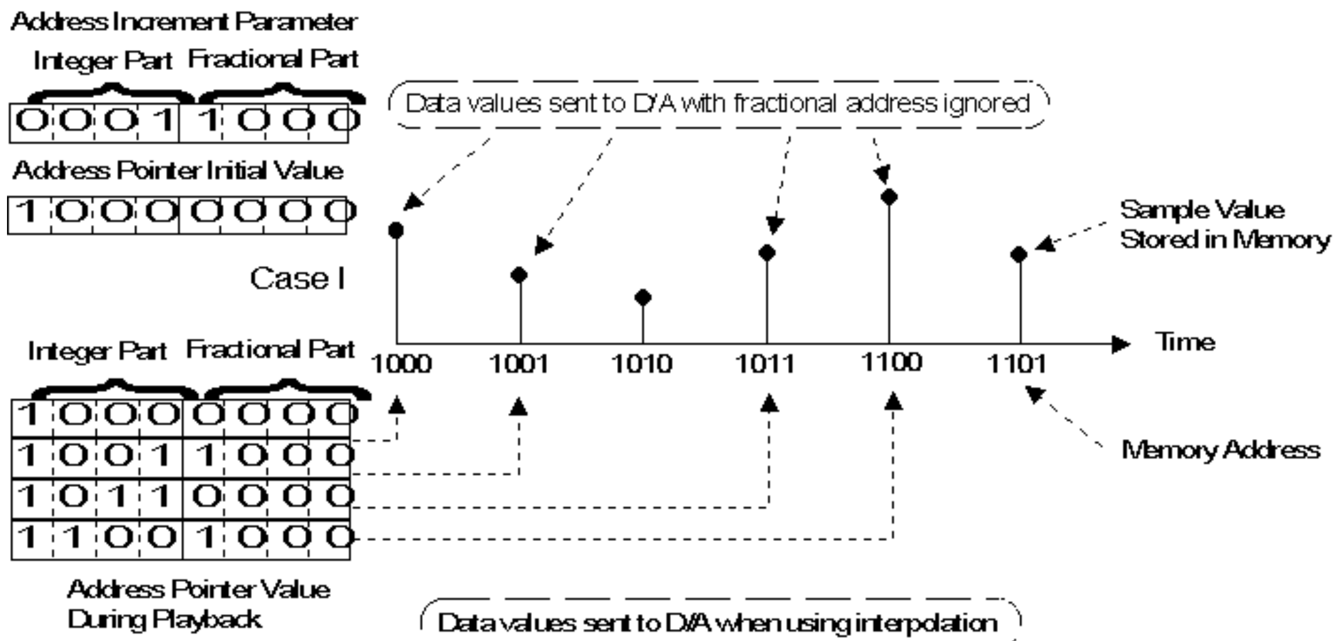
Wavetables (3)

- Keep a counter pointing to the sample being played (phase index) and an increment defining how much to add to the counter to reach the next sample (phase increment).
- Since memory positions are integer values, we can truncate or round increments - resulting on distortion

Phase Index	Truncated	error	Rounded	error
1	1	0	1	0
2.125	2	0.125	2	0.125
3.25	3	0.25	3	0.25
4.375	4	0.375	4	0.375
5.5	5	0.5	6	0.5
6.625	6	0.625	7	0.375
7.75	7	0.75	8	0.25
8.875	8	0.875	9	0.125
10	10	0	10	0
11.125	11	0.125	11	0.125
12.25	12	0.25	12	0.25
13.375	13	0.375	13	0.375
14.5	14	0.5	15	0.5

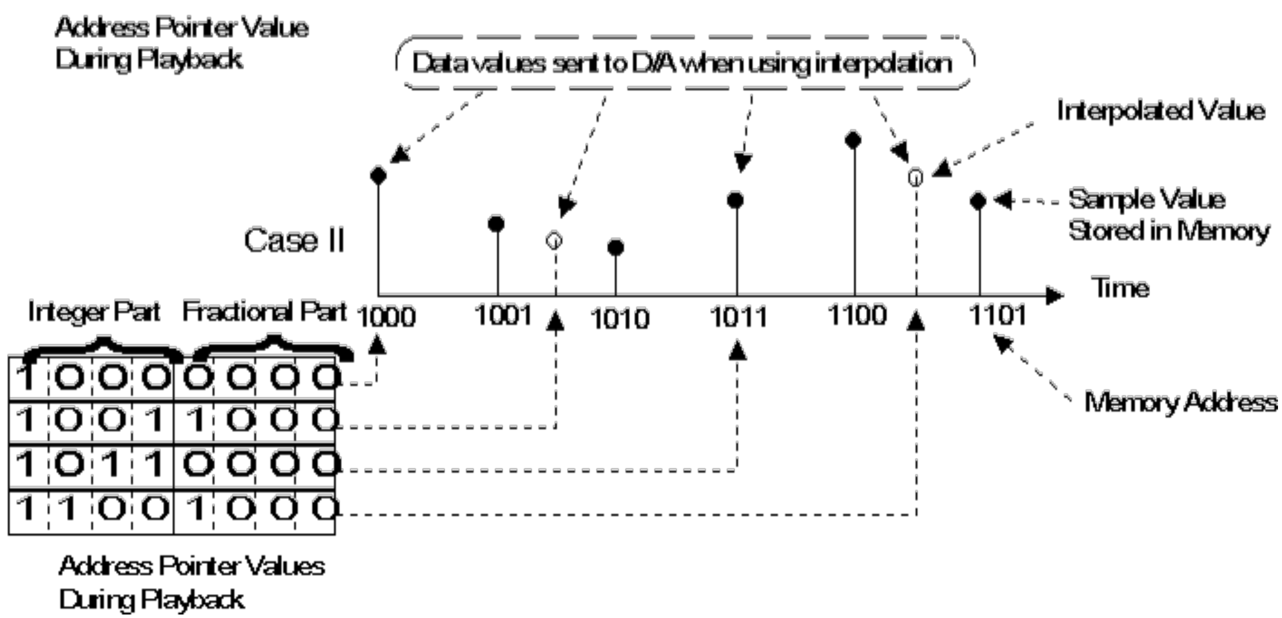
Wavetables (4)

- For accuracy, both counter and increment need to include a fractional part.
- The integer part of the phase index is to address the sample memory and the fractional part is to maintain frequency accuracy
- Frequency resolution is related to the number of bits of the fractional part



Wavetables (5)

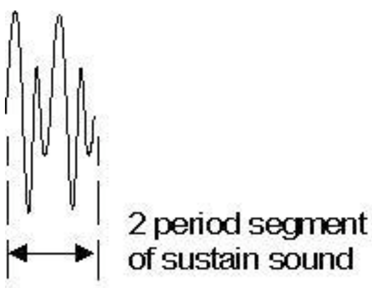
- A solution is to over-sample to increase frequency resolution: f_0 = (phase_increment * cycle * fs)/(L * oversampling factor)
- Requires large amounts of memory space
- Another solution is to interpolate: s(n+k)=(1-k)s(n)+(k)s(n+1)

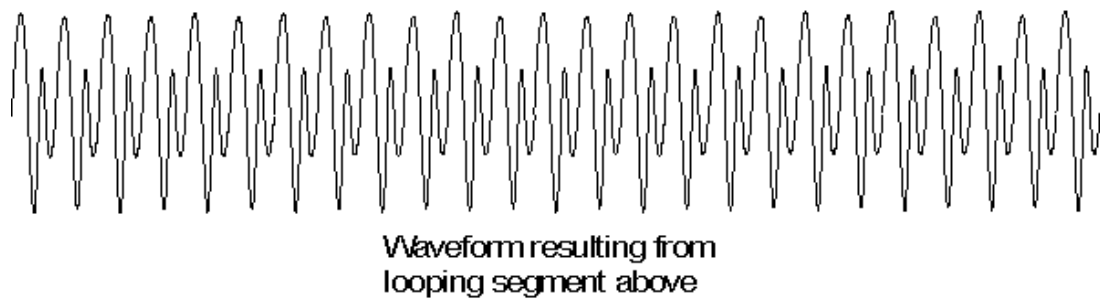


- Most implementations use a combination of both solutions

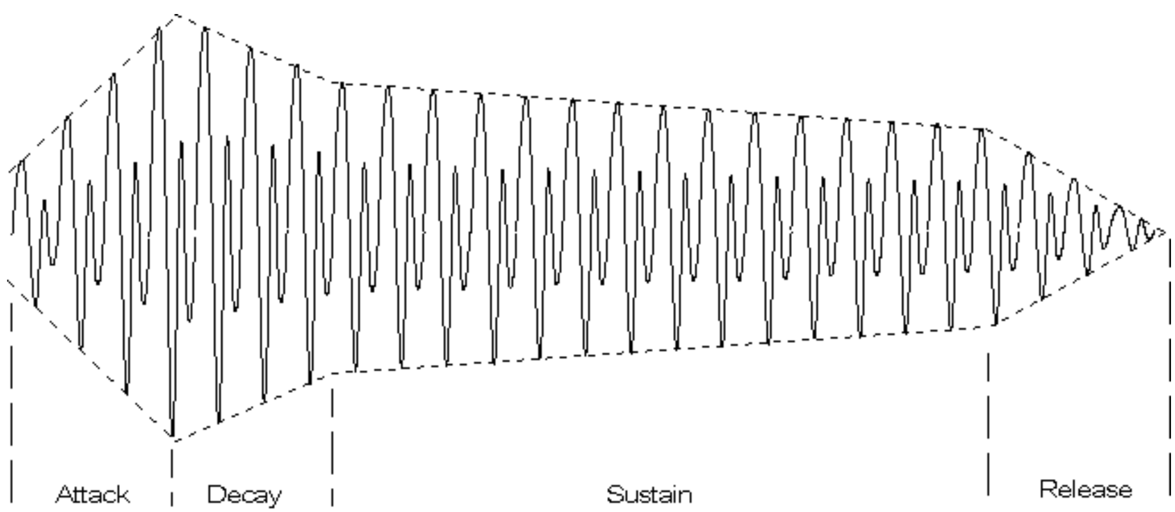
Wavetables (6)

- Continuous sounds are created by looping wavetables



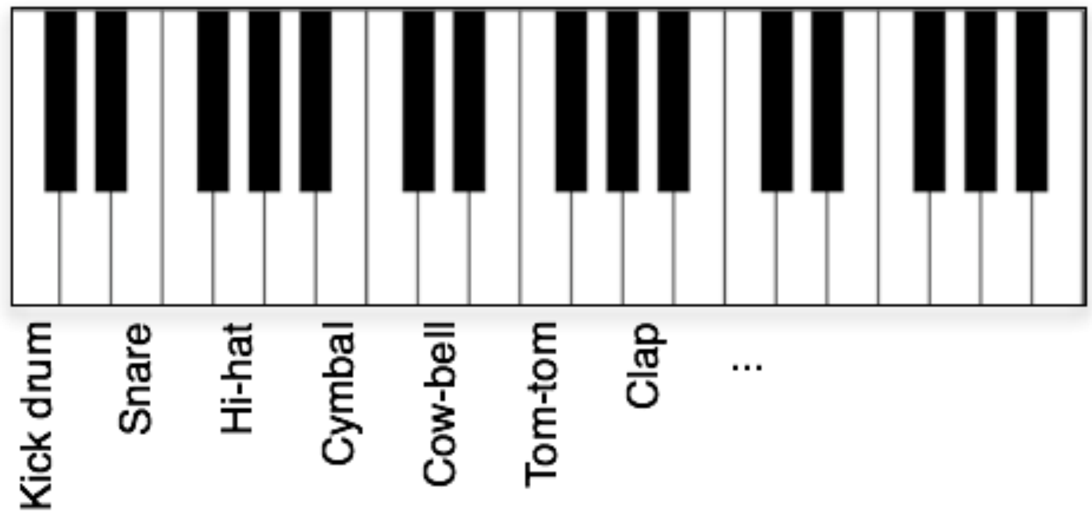


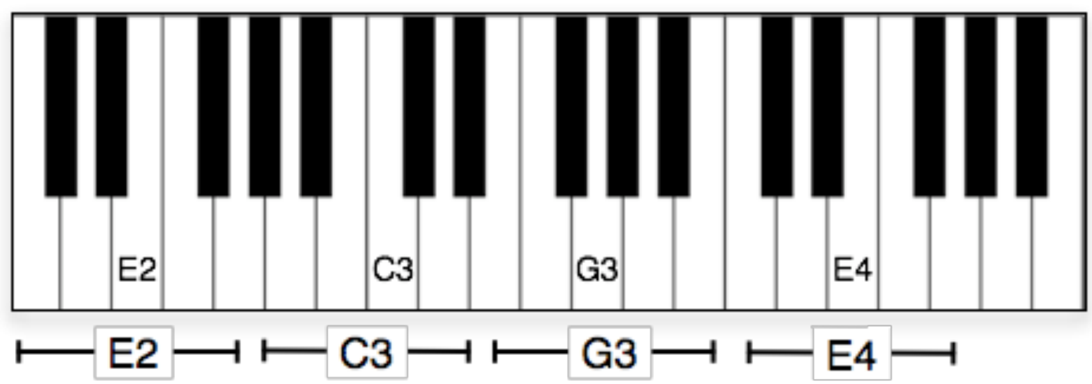
- Envelopes are used to make them time-varying:



Wavetables (7)

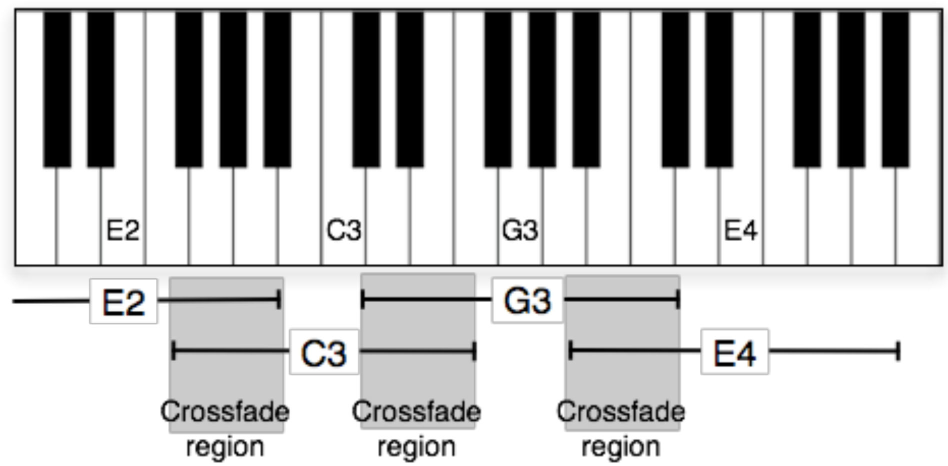
- A number of strategies can be used to assign sound samples to MIDI notes
- A sound sample per MIDI note (e.g. drum kit)
- A sound sample per MIDI note range: pitch shifting will be required to satisfy the pitch - note number relationship
- Less memory cost than previous approach

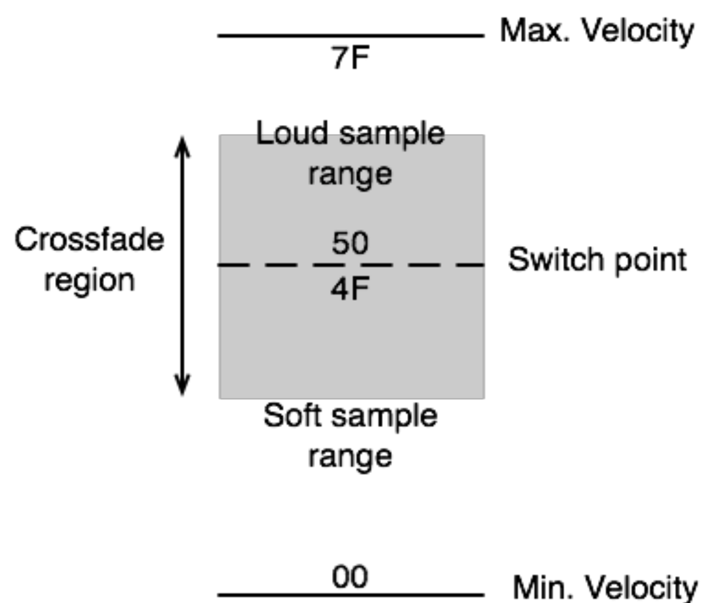




Wavetables (8)

- To avoid unnaturally sounding changes between samples, areas of crossfade can be used
- Pitch ranges for each sample are made to overlap.
- In the crossfade region a proportional mix of the two sounds is used
- Crossfades between loud and soft samples is also used (based on velocity)





Useful References

- Francis Rumsey and Tim McCormick (2002). "Sound and Recording: An Introduction", Focal Press.
  - Chapter 13: MIDI

- Joseph Rothstein (1995). "MIDI: A comprehensive introduction"
- MIDI Manufacturers Association (2002). The complete MIDI 1.0 detailed specification (www.midi.org)