Sample-based synthesis

Basic Ideas: Similar to subtractive synthesis or additive synthesis.

The **principal difference** is that the seed waveforms are sampled sounds or instruments instead of fundamental waveforms such as the saw waves of subtractive synthesis or the sine waves of additive synthesis.

Samplers, together with traditional Foley artists, are the mainstay of modern sound effects production.

Musical genres: Hip-hop, Trip-hop, Dance music, Garage, Jungle, Trance, Modern Electronica *invented* due to samplers.

Most music production now uses samplers.

Sample-based synthesis: Comparison with other Synthesis methods

- Advantages (over other methods of digital synthesis such as physical modelling synthesis (**more soon**) or additive synthesis): processing power requirements are much lower.
  - Nuances of the sound models are contained in the pre-recorded samples rather than calculated in real-time.
- Disadvantage: in order to include more detail, multiple samples might need to be played back at once
  - E.g. a trumpet might include a breath noise, a growl, and a looping soundwave used for continuous play
  - Reduces the polyphony as sample-based synthesizers rate their polyphony based on the number of multi-samples that can be played back simultaneously.

Sample-based synthesis: Examples

- Mellotron (analog tape) (1962)
- Computer Music Melodian (1976): Stevie Wonder's "Secret Life of Plants"
- CMI Fairlight (1979)
- NED Synclavier (1979).
- EMU Emulator series (1981)
- Akai S Series (1986)
- Korg M1 (1988): The M1 also introduced the "workstation" concept.
- Software Samplers (2005) : NI Kontakt, Steinberg Halion

Sampling Trivia

CMI Fairlight

- Cost the price of a good house (c. £20,000) when released in 1979.
- It is now available as an iPad App!
  - Fully functional

Sample-based synthesis Basics: Looping

A sample-based synthesizer's ability to reproduce the nuances of natural instruments is determined primarily by its library of sampled sounds.

Early days of Sampling (c. Late 1980s/Early 90s)

Computer memory expensive:

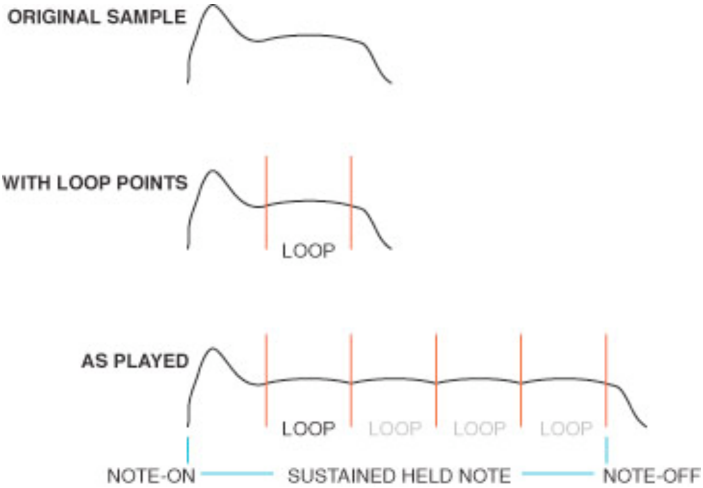
- Samples had to be as **short** and as **few** as possible.

This was achieved by **looping a part of the sample**

Looping today:

Looping still useful for

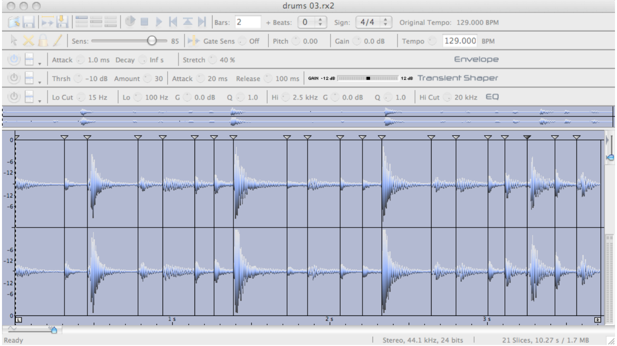
- Saving sample memory space — efficiency
- Looping audio material: Drum tracks, sound effects, *etc.*



Problem: How to find looping points?

Finding looping points

- Simple idea: Find silence points (**zero (amplitude) crossings**) in sample. E.g. Drum beats



  Loop between these

- Alternative: Find portions in sample that have same audio content — pattern matching.
  E.g. **Sustaining musical instruments**.

Sample-based synthesis Basics: Looping (cont)

Pitch control:

- Speed or slow up sample to change pitch (realism to only a few semitones in pitch change)
- Still need some sample across the range of the keyboard
- As memory became cheaper (and now with software based sample synthesis), it became possible to use **multisampling** — looping still used in individual samples.

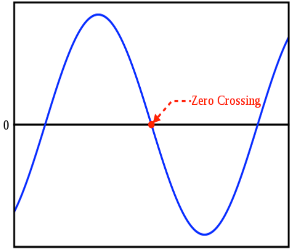
Finishing off the loop:

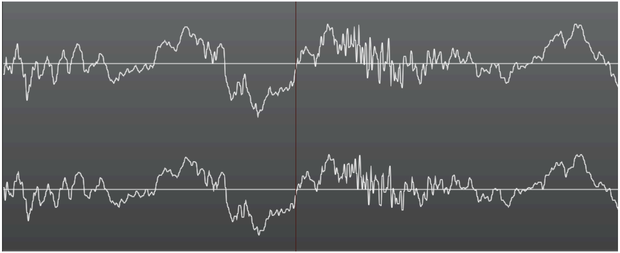
- Early Days: Use a volume envelope curve to make the sound fade away.
- Today: Include tail off sample in data — triggered by note off.

Beat Slicing Algorithms Background: Altering Loops

Silence Points:

Find silence points (**zero (amplitude) crossings**) in sample.
Snapping to silence points means that no nasty clicks in audio when joining audio together.
Too simple for many practical looping applications - **how to detect correct loop point**?





Beat Perception:

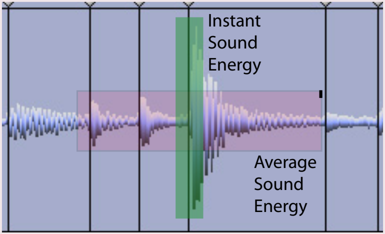
The human listening system determines the rhythm of music by detecting a pseudo — periodical succession of beats

- The more energy the sound transports, the louder the sound will seem.
- But a sound will be heard as a beat only if his energy is largely superior to the sound's energy history, that is to say if the brain detects a large variation in sound energy.
- Therefore if the ear intercepts a monotonous sound with sometimes big energy peaks it will detect beats,

Example of using Human Perception — a theme of this module.

Beat Slicing Algorithms Ideas (1):

Simple sound energy beat detection:



- Computing the **average** sound energy of the signal over a relatively large sample (around 1 second)
- Compute **instant** sound energy (around 5/100 second).
- Comparing average to the instant sound energy.
- We detect a beat only when the instant energy is larger than the local energy average.

Beat Slicing Algorithms Ideas (2):

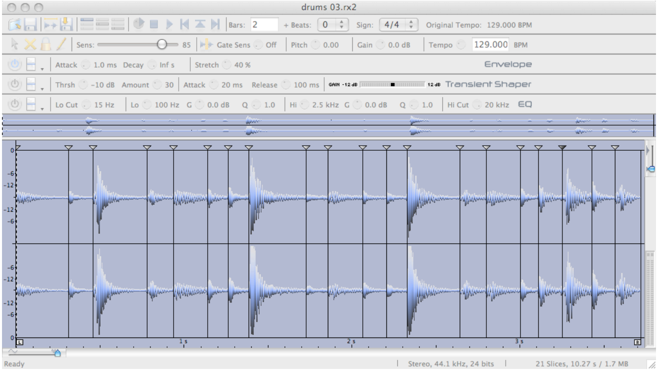
Frequency selected sound energy:

More elaborate model:

Try to detect large sound energy variations in particular frequency subbands

- Apply Fourier Transform — separate beats according to their frequency sub-band.
- Apply energy analysis but in frequency space:
    - Compute Fourier Transform over 1024 samples.
    - Divide into around 32 sub-bands.
    - Compute the sound energy contained in each of the subbands
    - Compare it to the recent energy average corresponding to this subband.
    - If one or more subbands have an energy superior to their average we have detected a beat.
- For more details search web or see references at end of section.

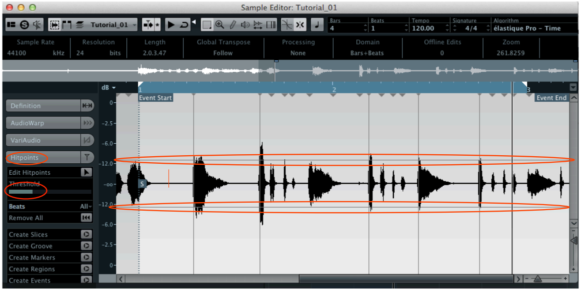
- Launch ReCycle and select and open a file.
- In the main window, you may click the Play button to hear the entire loop, from start to end (repeating until you click the Stop button.)
- **To create slices**: Adjust the Sensitivity slider to the right — the exact value depends on the audio.



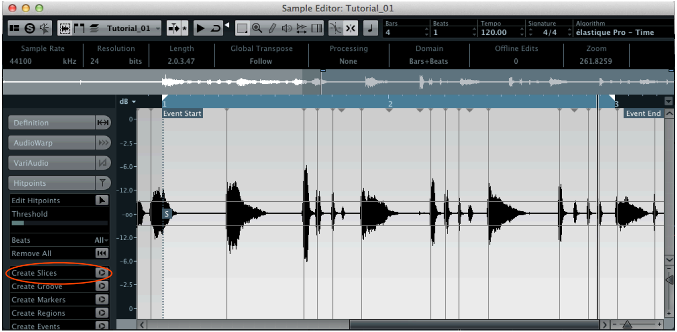
Cubase takes a simpler more user interactive approach to beat slicing.

In the **Sample Editor**

- Select **Hitpoints** Editing Option.
- Either adjust **Threshold** or visual **horizontal lines** to select the appropriate level of hit points, as displayed.



Beat Slicing in Cubase (Cont.)



- When happy hit the **Create Slices** button.
- Hitpoints can then be edited in a similar fashion to Recycle

Cubase Drum Sampler

Cubase has a built in Drum Sampler: **Groove Agent One**.

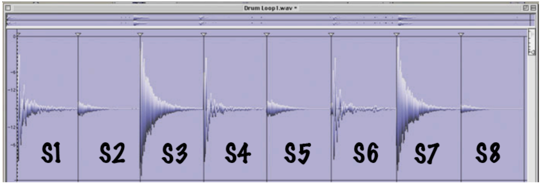
To map sliced beats (hit points) to Groove Agent One:

- Simply drag the sliced audio file onto one of the **Drum Pads**.
  - Subsequent slices are mapped to consecutive pads.

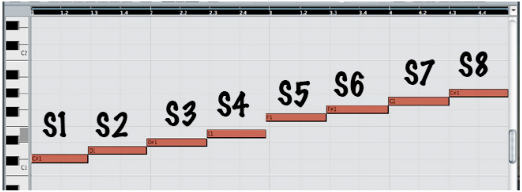
Beat Slicing — Simple Application

Recap: Create a MIDI performance of a chromatic scale, whose note timing trigger each sample at the perfect time to recreate the original audio.

Recycle Slicing:

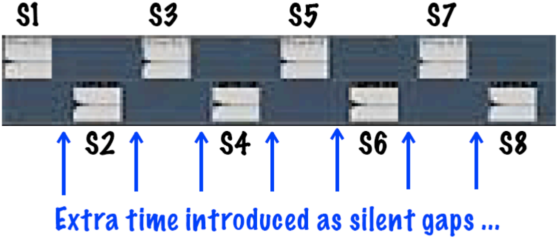


Midi Mapping/Triggering of Slices:

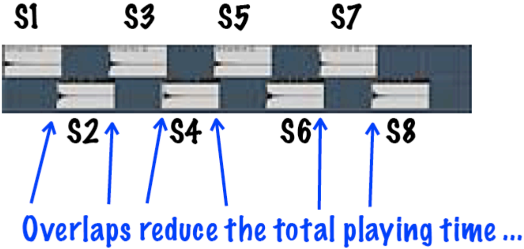


Beat Slicing — Tempo Change Problems

Replay after tempo is made slower:



Replay after tempo is made faster:



Beat Slicing - Artefacts, Solving the Tempo Problem

For drum loops etc. — **attacks** are artefact free.

- The most important part of a percussion sound.

Two artefacts (from previous slide):

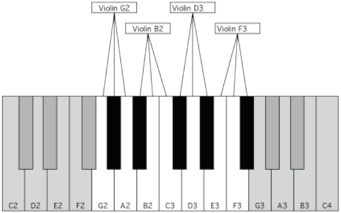
Beat Slicing Solution

Solutions

- Apply envelope to each slice to fade it to silence before gap or overlap.
- **For gaps**: loop the end of the tail to extend it through the gap.

Sample-based Synthesis: Multisampling

- Non-pitched simple example: the concept of drum mapping — **see also general MIDI section later**
- **Need to preserve relationships between key notes**
- **Multisampling Basic Idea**:
  - Sample instrument at regular intervals to cover regions of several adjacent notes (**splits**) or for every note.
  - **Advantage**: provides a more natural progression from the lower to the higher registers



Sample-based Synthesis: Example Kontakt Sampler Multisample Keymap

- When pluck a string or hit a drum or press a piano key, sound produced **depends** on how hard the action was.
- In software, this is measured by the velocity of a key press *etc.*
- Multisampling lays out samples vertically in keymap.
- **Velocity layers** layed out **horizontally**

(Single key mapped) Single Velocity Layer — Only **one** type of sound played at **any velocity**.

Volume output maybe controlled by velocity but **no change** in **timbre** of sound.

(Single key mapped) Dual Velocity Layer:

Sound **one** played at **lower level** velocity

Sound **two** played at **higher velocity**

Sample-based Synthesis: Velocity Layers (3)

(Single key mapped) Triple Velocity Layer — Three type of sounds played according to velocity.

Here upper velocity sound is being played.

Sample-based Synthesis: Key Map and Velocity Layers

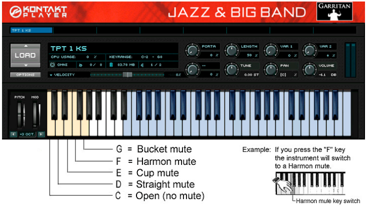
Most instruments are a combination of multisample key mapped and velocity layers

Sample-based synthesis Basics: Sample Keyswitching

- Instruments can make vastly different sounds depending how they are played
- Example: Trumpets (muted/not muted), violin (plucked, slow/fast up/down bow)
- For expressive performance samples can be **keyswitched**:

  Use keys (usually lower keys outside of instrument range) to select appropriate sounds

  - Essentially banks of key mapped velocity layered samples



Advanced Software Based Sampling

- Sampling now seems to have very few limits
- Full orchestras and <u>even choirs that can sing</u>
- Can sing words too (Advanced Keyswitching).
- Programming script control over sampler (Kontakt 2 and above).

Source: video

Sample Based Synthesis Further References

- *www.cs.berkeley.edu/ lazzaro/class/music209* — Good overview of Beat slicing. (I borrowed a few figures from here)
- Sound on Sound Magazine Beat Slicing Masterclass — *www.soundonsound.com/sos/jun04/articles/beatslicing.htm*
- *emusician.com/mag/square_one/emusic_slice/index.html* — Electronic Musician Magazine Article on Beat Slicing