Create a polyphonic drum-machine step-sequencer with different samples

Preparation. Remember that you can load wave files into tables using the object [soundfiler]. You need to create one table for each mono sound, and two tables for each stereo sound, i.e. one for each channel. Using mono sounds is utmostly recommended. The *-resize* command adjusts the size of the table automatically. Afterwards, You can send the *normalise* message to a table to adjust the content of your table so that *max_val = 1.*

The simplest way of reading audio from a table is using the object **[tabplay~]**, which accepts messages that tell them which part of the sound needs to be played (check the object's help to learn more about these messages).

It is expected that you use **encapsulation** (i.e. abstractions) to organise your patch. The fundamental idea is that you will create only one patch for all the "instruments", which then you should instantiate as many times as desired. **Remember that dollar signs can be used to assign creation arguments to abstractions.**

If you want to initialise some parameters, you can use **[loadbang]**, which sends a 'bang' when the patch is opened. This can be very useful to intialise numerical values, or load audio samples automatically.

Assignment steps and tips

1. Obtain the collection of samples to be used in your drum machine. You can use the 909 samples provided in this weeks' materials.
2. The following steps are to be applied to create a single "instrument" abstraction that will be called from a main patch as many times as desired:
    a. Load a soundfile into a table (this can be done automatically or manually).
    b. Prepare the patch to play the sound file via [tabplay~].
    c. Instead of using an individual [dac~] on each abstraction, use [throw~] to send all the audio signals to the main patch (there you should use [catch~] to collect all audio signals).

    d. Create a row of toogles to represent the individual steps of a single instrument: When toogles are ticked (1) the sound should play; when unticked (0) they should not produce sound
    e. Using [select] objects you can route the individual steps to the various toogles (as many as steps in your sequencer).
    f. Similarly, you can use additional select objects to trigger the individual sounds [select 1].
    g. Using the “graph-on-parent” (right click on white canvas) you can show the GUI objects in the specified area in the main patch. That would be extremely useful to manipulate your drum-machine from the main patch.
3. The following steps are to create and edit a **main** patch from where the “instrument” abstractions are called and general control parameters are set / send:
    a. I recommend you to instantiate a single “instrument” abstraction to develop your patches in parallel. Then you will make your instrument polyphonic simply by adding new instances of the “instrument” patch.
    b. Create a general tempo control in bpm to control the speed of your sequencer. The same tempo control via [metro] can be used to start/stop your instrument.
    c. Create a step-sequencer with a [hradio] to visualise the steps of the sequencer and send individual steps (integers) to the instrument abstractions. At this point you need to make the decission as to how many steps you might want: how many bars? how many divisions per beat?, etc. Typical step sequencers are powers of 2… e.g. 8 or 16 steps.
    d. Receive the audio signals from the “instrument” abstractions via a [catch~] object. Create a master volume control to control all instruments at once.

Add-ons

1. Add individual control volumes for each instrument
2. Can you imagina a way to add a pan control to each instrument too?