

Creative Coding (PAT 204/504, Fall 2024)

Lecture 20 – Drum Machines

Instructor: Hao-Wen Dong

Example 1: MIDI Drum Machine ("1_midi_drum_machine.maxpat")

• Use the "matrixcontrol" object to create a simple sequencer



- Click on the circles in lock mode to switch between 1 and 0
- Send a "getcolumn k" message to the "matrixcontrol" object to get the values of the k-th column as a list

>0	
	matrixcontrol
unpack 0 0 0 0 0. 1. 0. 1.	A "getcolumn k" message will return the values of the k-th column

- In the example above, we send a "getcolumn 0" message to the "matrixcontrol" object, and the output is "0., 1., 0., 1.", which is the first column (from top to bottom)
- Note that the "matrixcontrol" object is zero-based (i.e., the index starts from zero) by default

• Use several "select" objects to see if we get a one in a specific row



• Use a "metro" object to send a bang message every 200 ms and a "counter" object to count the bangs within a range, i.e., "0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, ..."



• Connecting everything gives us a simple sequencer interface



• Connecting the buttons at the bottom to a specific MIDI drum pitch gives us a simple MIDI drum machine



• We can do more than 16-beat rather than 8-beat drum patterns



• We can also do 6/8 times



Example 2: Sample-based Drum Machine ("2_sample_drum_machine.maxpat")

• Use a "buffer~" object to store an audio sample and a "play~" object to play the audio buffer



- In the example above, "buffer~ loop KID-BBapKit-Loop.wav" creates an audio buffer named "loop" that loads the "KID-BBapKit-Loop.wav" file at load time
- In the example above, "play~ loop 2" plays the audio buffer named "loop" with two channels (i.e., stereo sounds)

• Use multiple "buffer~" and "play~" objects to create a sample-based drum machine



Example 3: Synth Drum Machine ("3_synth_drum_machine.maxpat")

 Create a synth kick drum sound using a "cycle~" object with a low frequency and a "functiongraph~" based envelope



• Create a synth hihat drum sound using a "filtered noise" by filtering a pink noise signal (the "pink~" object) by a "filtergraph~" based lowpass filter and a

"functiongraph~" based envelope

Hihat			loadbang
<pre>dit_mode</pre>	2: hi 🜩	0	setdomain 500
	24 18 12 6 -6 -12 -12 -18		
biquad~		line~	
*~			
()			

• Use different oscillators, noise generators and filters to create the drum sounds

?	receive mysound loadbang	Snare	receive mysnare
<pre> edit_mode 2: hi </pre>	Setdomain 500		setdomain 500
24 18 12 6 6 12 6 12 18 19 19 19 19 19 19 19 19 19 19 19 19 19	Îne-	pink- biquad-	line-
Seriu- Signar		senu-signar	
Hihat	receive myhihat loadbang	Kick receive mykick loadbang	
Pink-	Secondari Coo	loadmess 50 50 cycle-	
send- signal		send- signal	

• Control the drum sounds using a "matrixcontrol" based sequencer using the "send" and "receive" objects to send and receive numbers or messages



• Use "send~" and "receive~" objects to send and receive signals



• Combing everything gives us a basic synth drum machines

metro 200 counter 7 petcolumn 51 2	Pinke sedomain 500 pinke sedomain 500 pinke sedomain 500 pinke sedomain 500 pinke sedomain 500 pinke sedomain 500 pinke	Snare (edt_mode 3: b (sidomain 500 pink- biguad- biguad- sidomain 500 pink- biguad- biguad- sidomain 500 pink- biguad- sidomain 500 pink- biguad- sidomain 500 pink- biguad- sidomain 500 pink- sidomain 500 pink- si
select se	Hihat Cadt_mode 2:hi0 Cadtang Cadtomain 500	Kick receive mysics casebang externain 500 9 50 Syde Une