Introduction to Algorithmic Composition and Pure Data

R. Yagiz Mungan
yagiz@purdue.edu

1) Algorithmic Composition

Algorithmic composition is creating music by using a set of steps. In a way, algorithmic composition is meta-composition, where you create a system, a set of rules or a function that creates music for you. Algorithmic composition can be deterministic or indeterminate, which it may or may not produce the same result.

One of the first examples of algorithmic composition belongs to Mozart and his "Musikalisches Würfelspiel", which is a dice game. In this game, dices determine instruments and music parts from a pool[1]. These music parts create the piece that will be performed.

Mostly, algorithmic composition is a field that combines computer science and music. However, in general algorithmic composition is an example of generative art. Instead of music, one can create text, images, videos, anything using a similar approach. In addition, the creation method can vary greatly and can be based on mathematics (statistics, probability and chaos theory), artificial intelligence, purely music theory or even chess[2].

The aim of algorithmic composition is not only to copy or mimic human behavior and tastes. But it also includes experimentation and the search for alternative approaches and tastes as well to find a deeper understanding of the existing theories. Algorithmic composition can be split into following types:

- Rule-based models
- Mathematical models
- Grammar-based models
- Evolutionary models
- AI-based models
- Hybrid models

The method that we will use today will be a part of the mathematical models. You will find algorithmic composition works in these fields: Music (both academic and popular), computer science, video games, art and cinema.

[1] http://sunsite.univie.ac.at/Mozart/dice/

[2] http://www.dada-companion.com/duchamp/music.php

2) Pure Data

Pure Data is a visual programming language. In Pure Data, instead of writing lines of code, one connects objects by wires. Pure Data is created for working with multimedia creation, editing, interaction and live performances in mind.

It is a free, open-source program that is used by many professionals. Pure Data can be downloaded from http://puredata.info/.

Pure Data operates with the idea of data flow, an object is triggered when it receives a signal and by wiring/patching objects together you create a data flow. An output of an object travels through the wires and triggers another object as a part of the data flow.

3) Pure Data Patches

Within the file package that you have found this document, you will also some files with .pd extension. These are Pure Data patches or codes. With this tutorial, I will show you how to create those patches. To start Pure Data in a Purdue University lab in Windows:

- Click on **Start**.
- Click on **All Programs**.
- Click on **Course Software** folder.
- Click on **technology** folder.
- Click on **cnit** folder.
- Click on **PureData 0.43.2**
- Let Pure Data load its libraries.

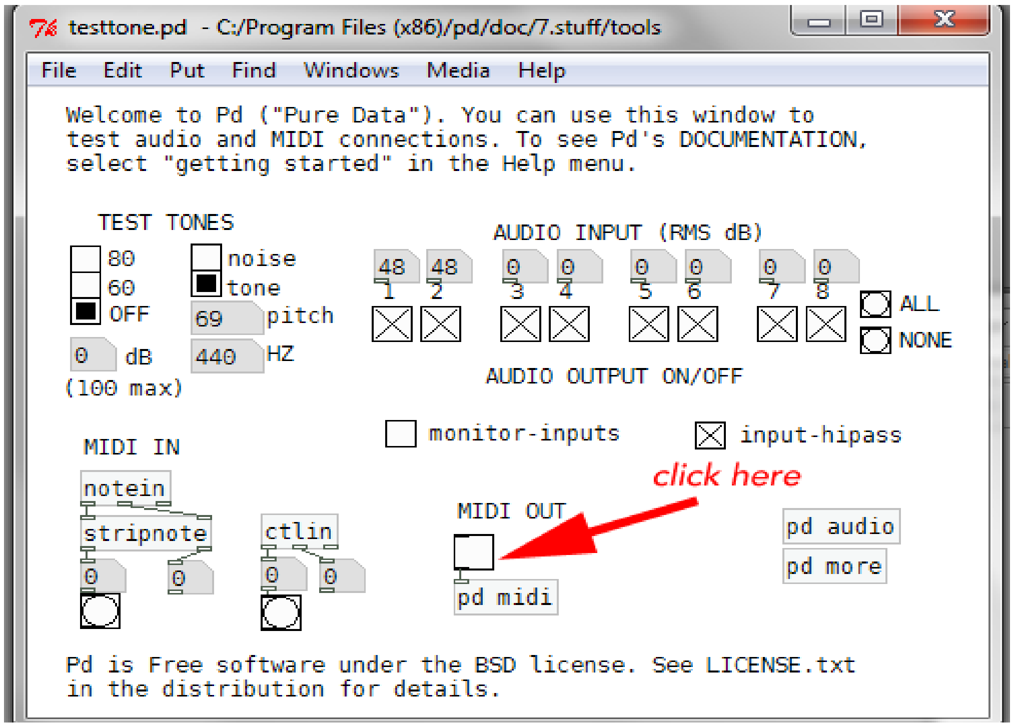
3.1 Setting up Pure Data

The first thing, we would like to do would be checking that the audio is set correctly.

- Start Pure Data.
- Go to **Media** tab.
- Click on **MIDI setting…**
- Click on **none** on the right of **output device 1:**
- Select **Microsoft GS Wavetable Synth.**
- Click **OK.**

Now, we have set a device for the MIDI output, if you have your own MIDI equipment you can select it there. The next step would be testing the MIDI connection.

- Go to **Media** tab.
- Click on **Test Audio and MIDI.**
- A new window will open; this is in fact a Pure Data patch.
- On the new window, click on the square under the **MIDI OUT** (see Figure 1).



Welcome to Pd ("Pure Data"). You can use this window to test audio and MIDI connections. To see Pd's DOCUMENTATION, select "getting started" in the Help menu.

Pd is Free software under the BSD license. See LICENSE.txt in the distribution for details.

Figure 1: Testing MIDI

- Now you should be able to hear sound, otherwise make sure that the computer is not muted. For Purdue University computers you might need to plug-in your head phones as the external speakers are disabled.
- **IMPORTANT:** Whenever you open a new Pure Data instance you need to set the output MIDI device again.

3.2 Creating Our First Note

Now, we will create our first MIDI not with Pure Data.

- After the steps in 3.1 are completed, go to **File** tab and click on **New.**
  - Shortcut for creating a new patch is **ctrl+n.**
- An empty Pure Data window should appear.
- Go to **Put** tab and click on **Object**
  - Shortcut for object is **ctrl+1**
- Click on the patcher screen to put “object”.

- Now we need to define what the object is. You will see that the object is blinking (if not double click on it) and write **makenote** and click an empty space**.**
- So far So far you should have something like in Figure 2.

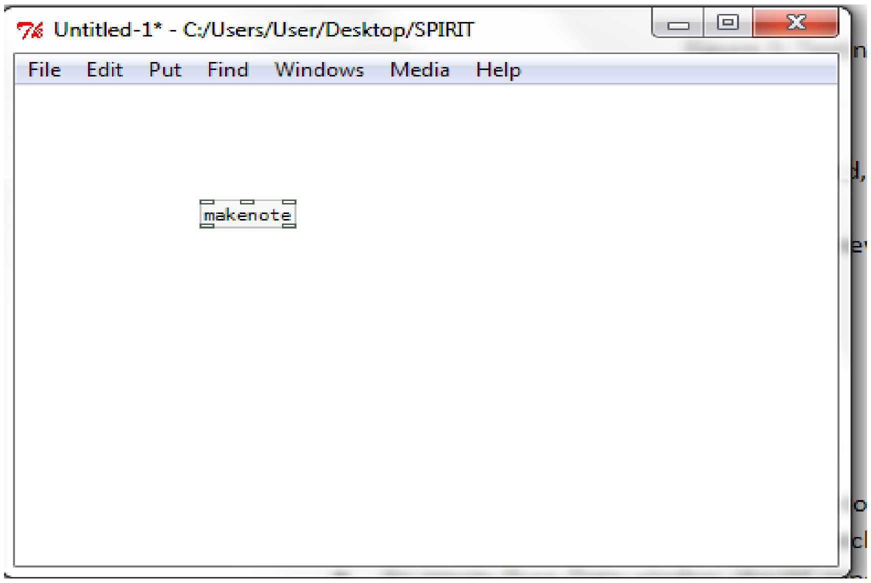

Untitled-1* - C:/Users/User/Desktop/SPIRIT
File Edit Put Find Windows Media Help
makenote

Figure 2: Makenote object

- You will see that **makenote** object has 5 nodes (the rectangles). Three on top and two at the bottom. The nodes on the top are the inputs; the nodes at the bottom are the outputs of the object.
- Now press **ctrl+1** to create another object. Put the second object slightly below the **makenote** object and label it as **noteout**.
- You will see that **noteout** only has three inputs and no outputs.
- Now hover the mouse over the first (left) output of **makenote** object, the cursor should become a circle.
- Click left mouse button and move the mouse around while keeping the left button clicked. You will see that a line (wire) appears.
- Now, move the wire to the first (left) node of **noteout**, and when the mouse icon becomes a circle release the left button.
- Now, you should see that the objects are connected (see Figure 3).
- Now, connect **makenote**'s second (middle) output to **noteout**'s second input.
- The **makenote** object takes three inputs and creates a MIDI note out of them. The **noteout** object takes the note and sends it to the MIDI device we set earlier at 3.1.
- Now we will configure the inputs for the **makenote**.
- Go to **Put** tab and click on **Number2.**
    - Shortcut for number2 is **ctrl+shift+n**.
- Place the number object above the **makenote** object.
- Put two more **number2** objects.

    - Alternately you can copy paste the one we have just put. You can use **ctrl+c** and **ctrl+v** as you normally do.
- Connect the outputs of the number objects to each input of the **makenote** object (see Figure 4).

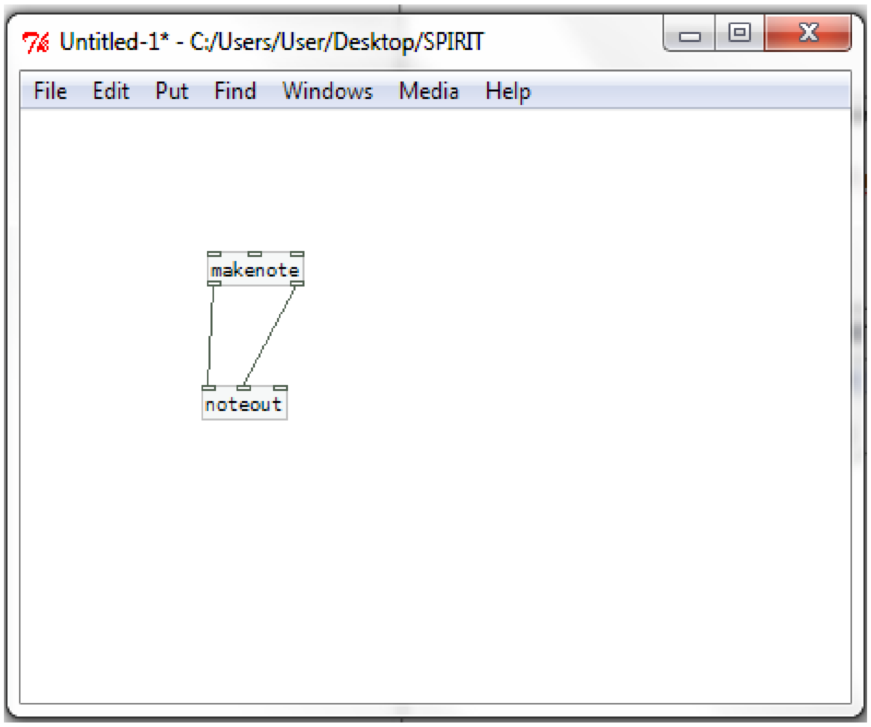

Untitled-1* - C:/Users/User/Desktop/SPIRIT
File Edit Put Find Windows Media Help
makenote
noteout

Figure 3: Makenote and noteout connected

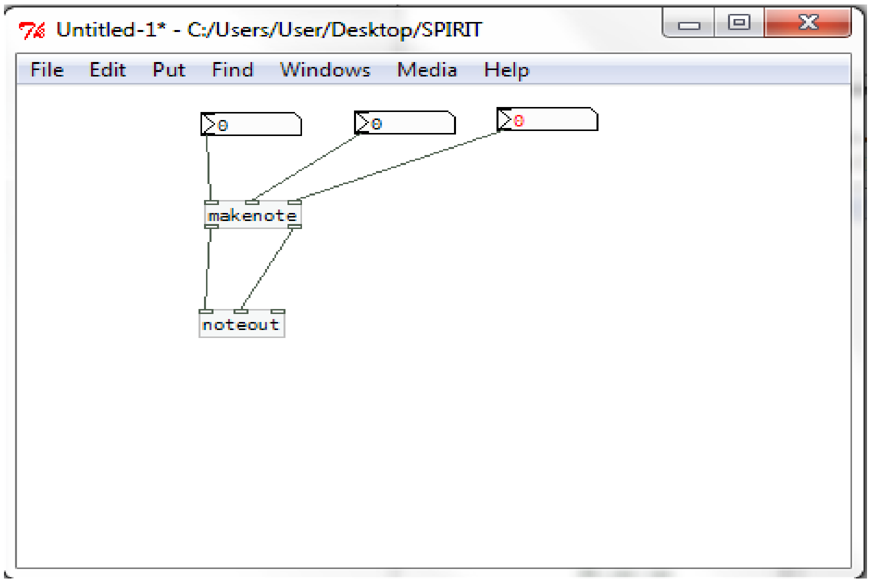

Untitled-1* - C:/Users/User/Desktop/SPIRIT
File Edit Put Find Windows Media Help
0
0
0
makenote
noteout

Figure 4: Three number objects connected

- The first input for the **makenote** is the pitch in MIDI (not actual frequency).
    - This can be a number between 0 and 255, where 0 is the lowest in frequency also.
- The second input for the **makenote** is the velocity in MIDI, which is the loudness of the volume.
    - This can be a number between 0 and 127, where 0 is silent.
- The third input for the **makenote** is the duration expressed in milliseconds (thousandth of a second).

So far our patch has been in the **edit mode**, where we could put new objects but could not activate anything.

- Click on **ctrl+e** to leave the edit mode. The cursor should change from the hand to an arrow.
- Click on the third number (the duration) and move the mouse up while holding the click you should see that the value in the number box is increasing.
- Repeat this process till you reach around 250.
- With the same technique increase the second number to around 100.
- Now, apply the same to the first one and you will hear the sound.

Press **ctrl+shift+s** or go to **File** tab and click on **Save As** to save as your patch. Choose a folder (your "Home Directory" is a good choice at Purdue University computers) and name it as **3_2**.

- **IMPORTANT:** To learn what an object does, in the **edit mode**, right click on the object and select **Help**. You can also go to properties and make changes (such as you can put a limit to the number inputs) by right clicking on an object.

3.3 Random Music

Our first algorithmic composition will be playing random notes with fixed loudness and duration (somebody hitting random keys mindlessly/mechanically on a piano). You can continue on the patch you have created for 3.2.

- Press **ctrl+shift+s** or go to **File** tab and click on **Save As** to save as your patch. Choose a folder and name it as **3_3**.
    - Do not forget to save your patch frequently.
- Enter **edit mode** by pressing **ctrl+e.** In **edit mode**, the mouse cursor is a hand.
- Press **ctrl+1** to put a new object above **makenote**. Label the object as **random.**
- **Random** object creates a random number.
    - It has two inputs. With the right input, you can set the maximum limit for the random number. The left input, you can send a trigger signal that activates the **random** object.

In Pure Data, the general, default trigger signal is "**Bang"**. Now we will put an object that sends a bang.

- Go to **Put** tab and click on **Bang.**
    - Shortcut for **Bang** is **ctrl+shift+b.**
- Put the bang object just above the **random** object.
- Select the wire that connects the first **number** object to the **makenote**'s leftmost input. The icon should become an "x". Now hit "delete" to remove that wire.
- Connect the **number**'s output to the right input of the **random** object.
- Connect the output of the **bang** object to the left input of the **random** object.
- Connect the output of the **random** to the left most input of the **makenote** object.
- Press **ctrl+e** to leave the edit mode.
- Set the first number to 200, second to 100 and the third to 250 approximately (see Figure 5).

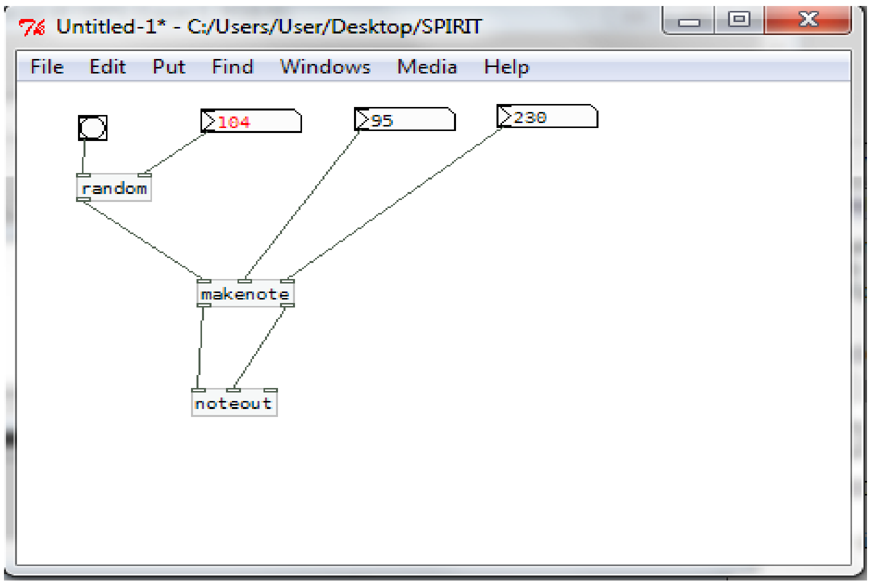


Figure 5: Random and Bang connected.

- Click on **bang** multiple times to hear different sound each time.
- Press **ctrl+e** to enter the edit mode.
- Press **ctrl+1** to put a new object just under **random**. Label the object as **print.**
- Connect the output of the **random** to input of **print,** while leaving the other connection intact.
    - There can be multiple connections within a node.
- Press **ctrl+e** to leave the edit mode.
- Click on **bang** multiple times.
- Now go to the main Pure Data window (You can press **ctrl+r** if you can not find it).

- On the console you will see "**print:** " followed by some random number. That number is the result of the **number** object. You can and should use print object for tracing, debugging and fixing the patch and the signals.

So far our system is still not automated. We need something that produces **bang** signals automatically.

- Press **ctrl+e** to enter the edit mode.
- Press **ctrl+1** to put a new object above **random**. Label the object as **metro.**
  - **Metro** is a metronome that sends **bang** signals periodically.
- Delete the wire between **bang** and **random**.
- Move **bang** above **metro.**
- Press **ctrl+shift+n** to create another **number** object. Put the object above **metro**.
- Now connect **bang** to **metro**'s left input and the new **number** to **metro**'s right input. Connect metro's output to **random**'s left input (see Figure 6).
- **IMPORTANT:** As you might realize, things can easily get messy so take some time to order your patch.

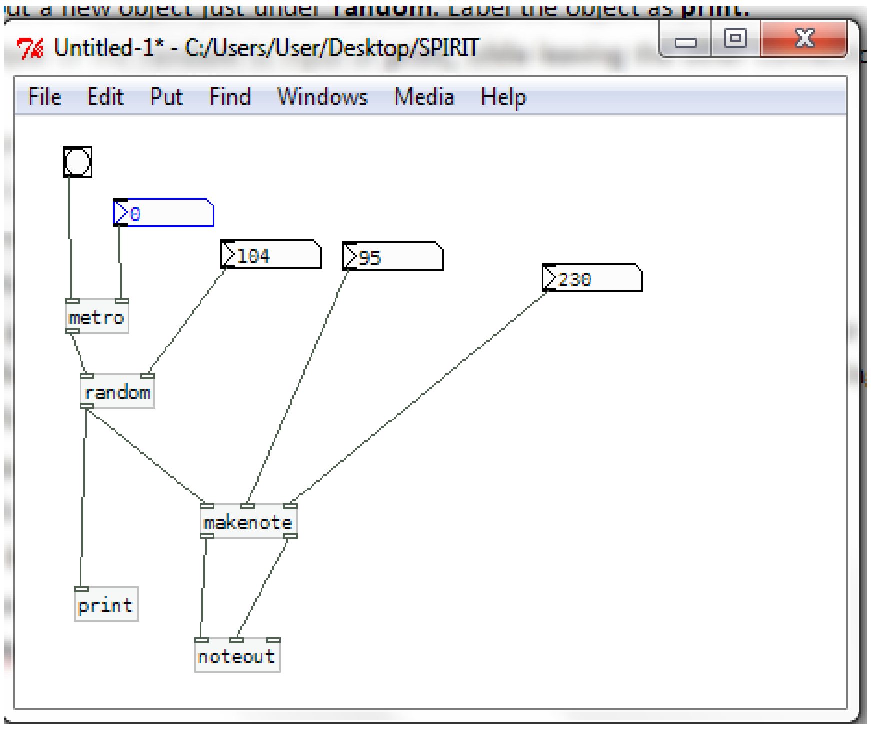


Figure 6: Metro connected

- Press **ctrl+e** to leave the edit mode.
- Increase the new **number** value to around 250.
- Click on **bang** once and it will go on forever.
  - Feel free to play with the **metro** period, **velocity** and **duration.**

- Now, we need something to stop the **metro.**
- Press **ctrl+e** to enter the edit mode.
- Go to **Put** tab and click on **message**.
    - Shortcut for message is **ctrl+2.**
- Put it somewhere above the **metro**, and label it as **stop**.
- **IMPORTANT:** Observe that it has a slightly different shape than the other objects because it is a **message**. In Pure Data, messages are used to set the objects or send commands to the objects.
- Connect the output of **stop** to **metro**'s left input.
- Press **ctrl+e** to leave the edit mode.
- Click on the **stop message** to stop the metro object.

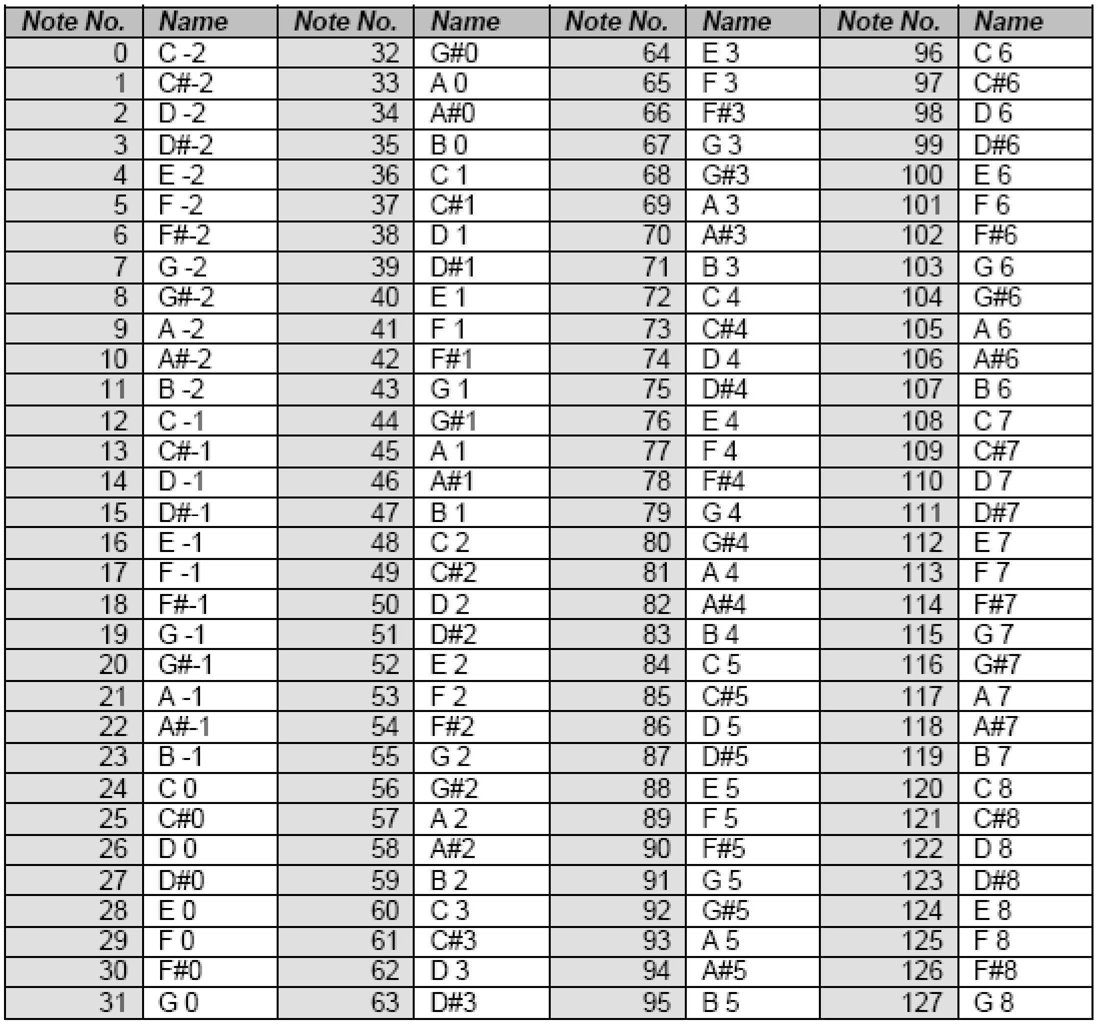
You have successfully created your first algorithmic composition. Do you like music?

3.4 Random within Limited Frequency

One quick and simple upgrade, we can make to the patch in 3.3, is to limit the frequency. In general, human ears can hear 20 Hz to 20000 Hz. However, not all the regions are heard with the same perceived loudness.

Now we will limit the random number so that it does not create very high or very low notes. Figure 7 shows the MIDI notes and the actual note they correspond. Looking at the Figure, let's limit our frequency from C2 to C6. The corresponding MIDI values are 48 and 96.

- Press **ctrl+shift+s** to save as your patch. Name it as **3_4**.
- Press **ctrl+e** to enter the edit mode.
- Click or double-click on **random** and change it to **random 48.**
    - This gives us a **random** object with an upper limit of 48 as the default value.
    - We could use the same approach to give **metro** a default value.
- Press **ctrl+1** to put a new object. Label that object **+ 48**. Put it below random.
    - This is a **+** object with default value of 48 for the second input.
    - **+** object does addition.
- Delete the wire between **random** and **makenote.**
- Connect **random**'s output to **+ 48**'s left input.
- Connect **+ 48**'s output to **makenote** (see Figure 8)**.**
- Delete the wire between **random** and **print.**
- Connect the output from **+ 48** to **print.**
- Press **ctrl+e** to leave edit mode.
- Click on **bang** to start. Now, all the notes should be in the easily hearable region. You can also observe the values from the console by pressing **ctrl+r.**
    - Feel free to experiment with other values.



Note No.	Name	Note No.	Name	Note No.	Name	Note No.	Name
0	C -2	32	G#0	64	E 3	96	C 6
1	C#-2	33	A 0	65	F 3	97	C#6
2	D -2	34	A#0	66	F#3	98	D 6
3	D#-2	35	B 0	67	G 3	99	D#6
4	E -2	36	C 1	68	G#3	100	E 6
5	F -2	37	C#1	69	A 3	101	F 6
6	F#-2	38	D 1	70	A#3	102	F#6
7	G -2	39	D#1	71	B 3	103	G 6
8	G#-2	40	E 1	72	C 4	104	G#6
9	A -2	41	F 1	73	C#4	105	A 6
10	A#-2	42	F#1	74	D 4	106	A#6
11	B -2	43	G 1	75	D#4	107	B 6
12	C -1	44	G#1	76	E 4	108	C 7
13	C#-1	45	A 1	77	F 4	109	C#7
14	D -1	46	A#1	78	F#4	110	D 7
15	D#-1	47	B 1	79	G 4	111	D#7
16	E -1	48	C 2	80	G#4	112	E 7
17	F -1	49	C#2	81	A 4	113	F 7
18	F#-1	50	D 2	82	A#4	114	F#7
19	G -1	51	D#2	83	B 4	115	G 7
20	G#-1	52	E 2	84	C 5	116	G#7
21	A -1	53	F 2	85	C#5	117	A 7
22	A#-1	54	F#2	86	D 5	118	A#7
23	B -1	55	G 2	87	D#5	119	B 7
24	C 0	56	G#2	88	E 5	120	C 8
25	C#0	57	A 2	89	F 5	121	C#8
26	D 0	58	A#2	90	F#5	122	D 8
27	D#0	59	B 2	91	G 5	123	D#8
28	E 0	60	C 3	92	G#5	124	E 8
29	F 0	61	C#3	93	A 5	125	F 8
30	F#0	62	D 3	94	A#5	126	F#8
31	G 0	63	D#3	95	B 5	127	G 8

Figure 7: MIDI notes

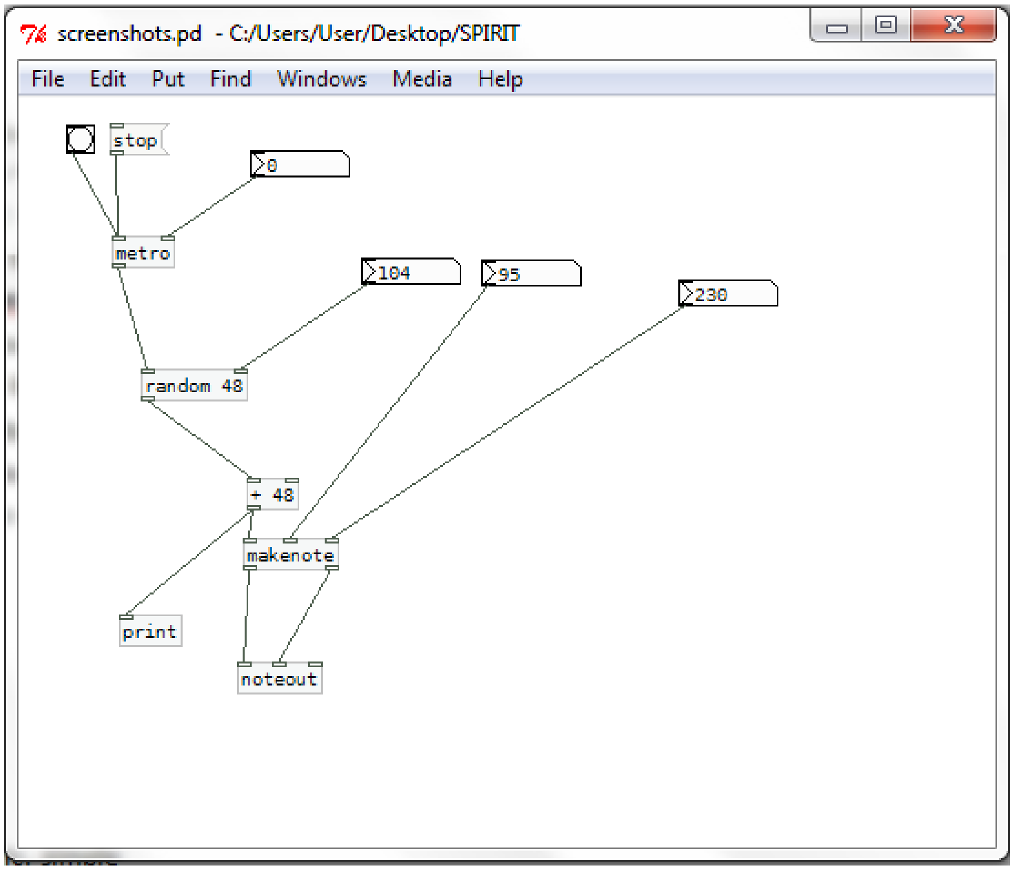

screenshots.pd - C:/Users/User/Desktop/SPIRIT
File Edit Put Find Windows Media Help
stop
0
metro
104
95
230
random 48
+ 48
makenote
print
noteout

Figure 8: Random within limited frequency

3.5 Random with Limited Step Size

You will realize that the random music just jumps around and most people does not like it. The reason is because it is not natural, for humans it is hard to make large pitch leaps while singing and same is true for most instruments.

This time we will limit the change so that the next note can only be certain distance away from the current one. For this setup, we will use an object named **drunk**. **Drunk** object gives us a random number that is within a defined distance around a starting point.

- Press **ctrl+shift+s** to save as your patch. Name it as **3_5**.
- Press **ctrl+e** to enter edit mode if you are not in edit mode.
- Click on **random** and rename it as **drunk.**
  - This removes the random object and replaces it with drunk
- **Drunk** has three inputs. Instead of sending each input independent values we will send all the three together.
- Delete the **number** connected to **drunk.**

- Press **ctrl+2** to put a **message** above drunk.
- Inside the message write **24 48 4.**
    - Remember for the random we have chosen the MIDI note range 48-96. We will use the same range.
    - 24 is our starting point, which is also the middle value.
    - 48 is the maximum number.
    - 4 is the maximum step size. The step can be either positive or negative. So drunk will output numbers within plus/minus 4 range starting with 24.
- Connect the output of the **message** to **drunk**'s left most input (see Figure 9).

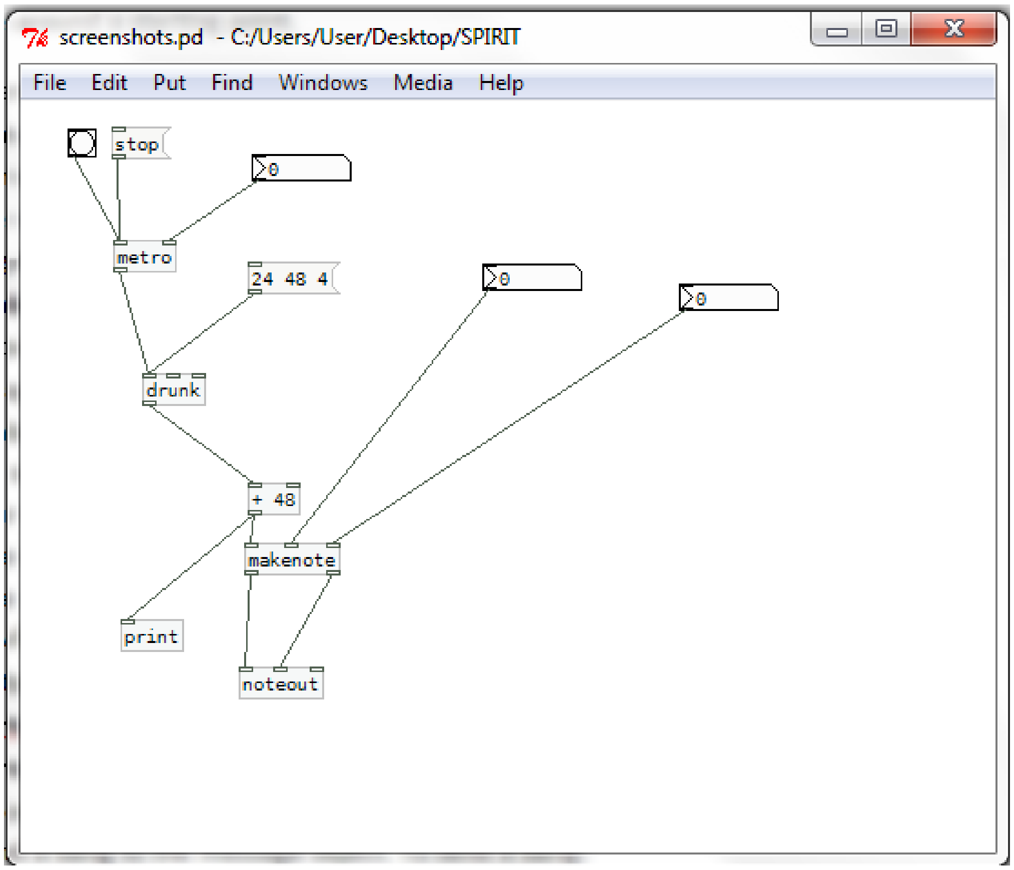


Figure 9: Drunk connected

- Now press **ctrl+e** to leave edit mode.
- Adjust the **metro** period, **makenote** velocity and **makenote** duration as we did before.
- **IMPORTANT:** Connecting a message object does not send the message to its target. You also have to send a bang to the message object. To send a bang:
    - You can put a **bang** object and use it.
    - You can click on the **message** (when not in edit mode).
    - You can also use **loadbang** which sends a bang when a patch is open. However, putting a **loadbang** does not bang (it only and only sends the bang when a patch is opened).

    - Whenever you open a patch the **message** objects needs to be reinitialized, thus putting **loadbang** objects will help you a lot in the long run.
- Click on the **message** to send it to **drunk.**
- Activate the **metro** object.

3.6 Random/Drunk within a scale

Next step, in enhancing the melody would be limiting the notes that we use. A scale is a set of notes that follow a certain distance pattern. The first two examples for scales are the major and the minor scale. Here we will limit our notes to C major scale (which is the white keys of the piano). In C major scale the notes are: C – D – E – F – G – A – B

Looking at Figure 7, we see that within our range (48-96) the C major notes are: 48 50 52 53 55 57 59 60 62 64 65 67 69 71 72 74 76 77 79 81 83 84 86 88 89 91 93 95 96.

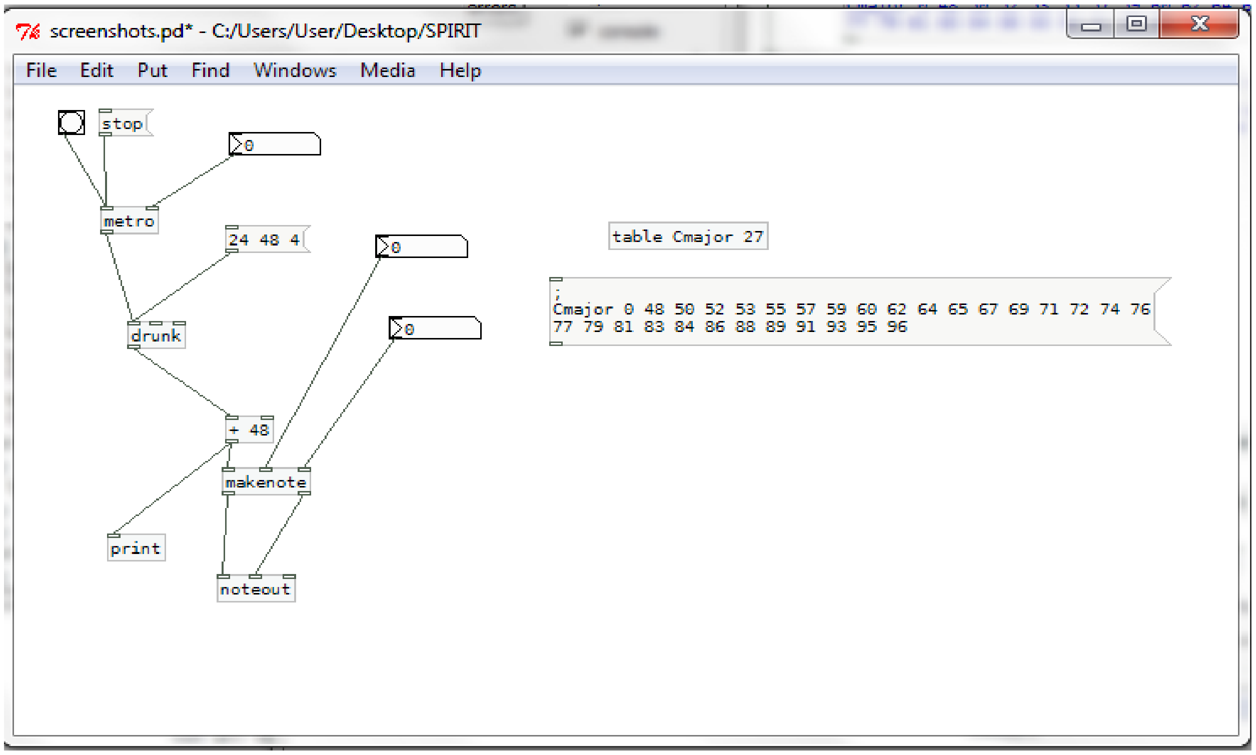
The plan is to create an **array** with these values and use our **random** or **drunk** object to choose a number from that array.

- Press **ctrl+shift+s** to save as your patch. Name it as **3_6**.
- Now press **ctrl+e** to enter edit mode.
- Press **ctrl+1** to create an object, and write **table Cmajor 27**.
    - Here we are creating a **table** object. Table is a type of array in Pure Data.
    - The name of the table is **Cmajor.**
    - The length of the table is 27.
- Now we will fill the **table**. Press **ctrl+2** to create a message and copy paste the following into the message:

;

Cmajor 0 48 50 52 53 55 57 59 60 62 64 65 67 69 71 72 74 76 77 79 81 83 84 86 88 89 91 93 95 96

- The format of the message is as follows:
    - "**;**" (semicolon) followed by **return button** "enter"
    - **Name of the array/table**
    - The index of the array/table that you want to start writing
    - The values that you want to write.
- Thus the 0th element will be 48, the 1st will be 50 and so on.
- Since this is a **message** you need to send it. You do not need to connect it but you need to leave the edit mode via **ctrl+e** click on the **message** and enter the edit mode again via **ctrl+e**.
- Now, the patch should look like Figure 10 if you are continuing from the previous section.

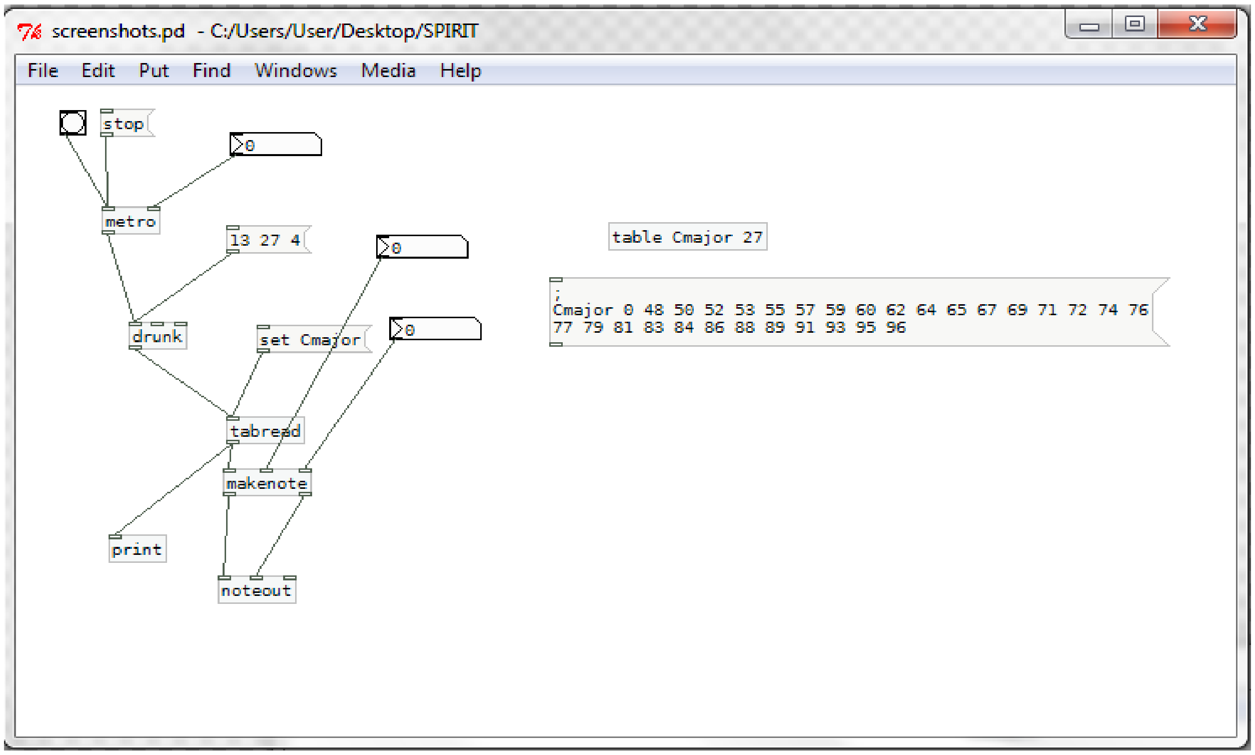

screenshots.pd* - C:/Users/User/Desktop/SPIRIT
File Edit Put Find Windows Media Help
stop
0
metro
24 48 4
0
0
table Cmajor 27
drunk
+ 48
makenote
print
noteout

```
;
Cmajor 0 48 50 52 53 55 57 59 60 62 64 65 67 69 71 72 74 76
77 79 81 83 84 86 88 89 91 93 95 96
```

Figure 10: Cmajor table

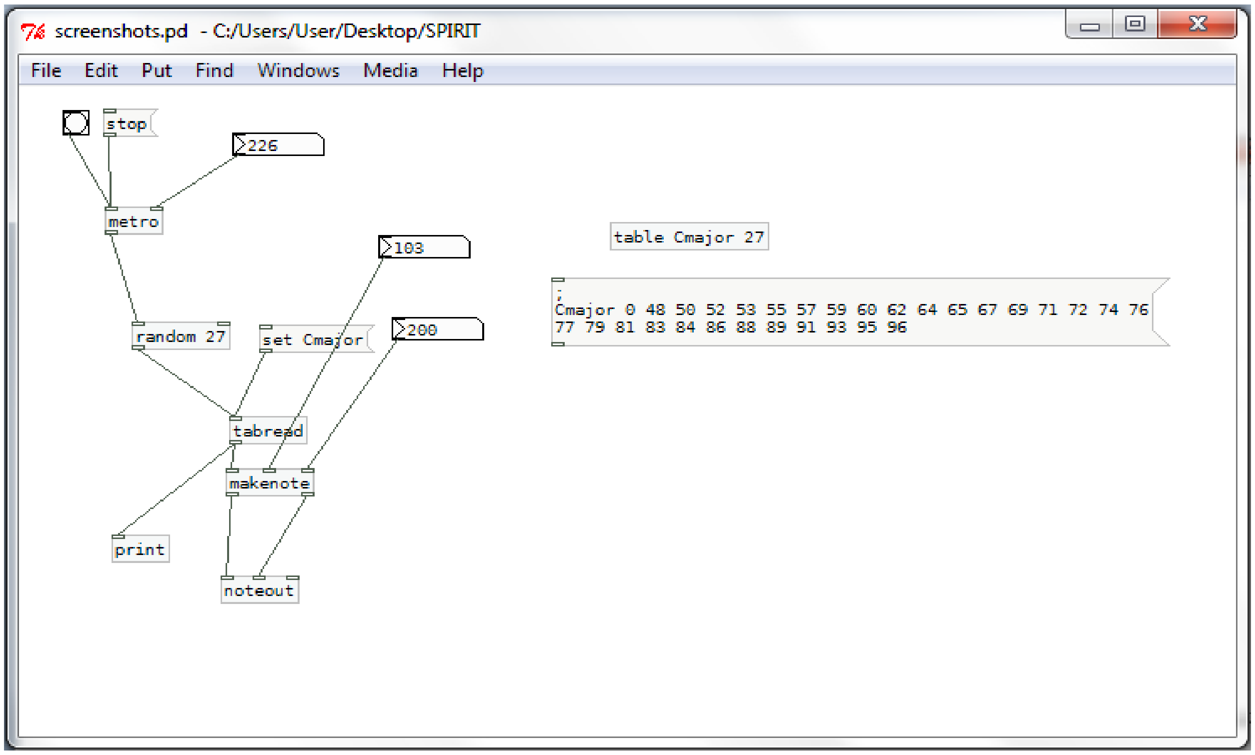
- Now press **ctrl+e** to enter edit mode.
- Now if we want to use the **drunk** object we need to change its parameters. So click on the **message** that says “24 48 4” and change it to “13 27 4” since our table has 27 elements.
- Since this is a **message** you need to send it. Leave the edit mode via **ctrl+e** click on the **message** and enter the edit mode again via **ctrl+e**.
- Delete **+ 48** since the data we will use is already raised to 48.
- We will use **tabread** object to read from the **table.** Press **ctrl+1** to create an object, put it where **+ 48** was and name it **tabread.**
  - **Tabread** needs a message to define, which table/array it will read.
  - **Tabread** takes a numerical input which will be the index of the item that it will output.
- Press **ctrl+2** to create a message, put it above **tabread** and write **set Cmajor** inside it.
- Connect the output of the **set Cmajor** message to the input of the **tabread** object.
- Since this is a **message** you need to send it. Leave the edit mode via **ctrl+e** click on the **message (set Cmajor)** and enter the edit mode again via **ctrl+e**.
- Connect the output of the **tabread** to the input of the first input of **makenote** and **print** object.
- Connect the output of the **drunk** object to the input of the **tabread** object. Your patch should look like Figure 11.
- Set up the metronome period, velocity and duration as usual and start via the **bang**.

- Instead of **drunk** you can use **random 27** as in Figure 12, for a random melody within C major scale.


screenshots.pd - C:/Users/User/Desktop/SPIRIT
File Edit Put Find Windows Media Help
stop
0
metro
13 27 4
0
drunk
set Cmajor
0
tabread
makenote
print
noteout
table Cmajor 27

```
;
Cmajor 0 48 50 52 53 55 57 59 60 62 64 65 67 69 71 72 74 76
77 79 81 83 84 86 88 89 91 93 95 96
```

Figure 11: Tabread connected


screenshots.pd - C:/Users/User/Desktop/SPIRIT
File Edit Put Find Windows Media Help
stop
226
metro
103
random 27
set Cmajor
200
tabread
makenote
print
noteout
table Cmajor 27

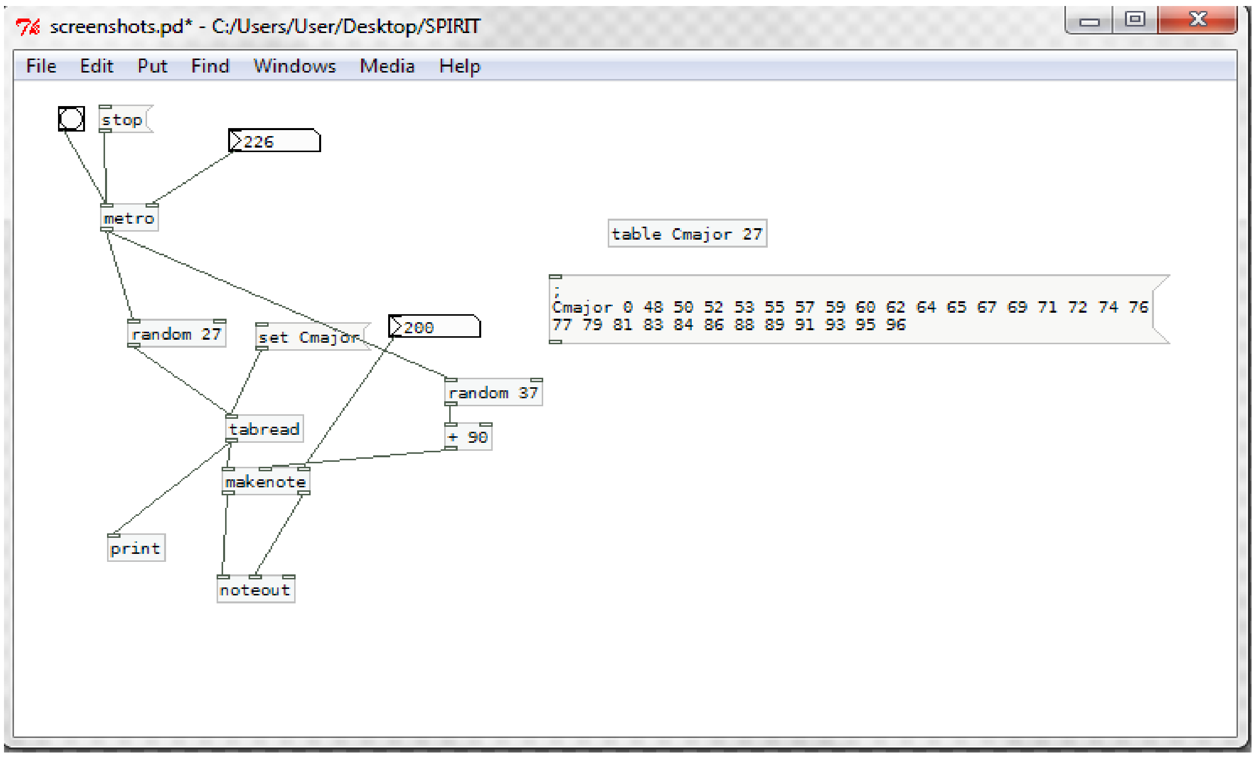
```
;
Cmajor 0 48 50 52 53 55 57 59 60 62 64 65 67 69 71 72 74 76
77 79 81 83 84 86 88 89 91 93 95 96
```

Figure 12: Random within C major

3.7 Dynamics

So far we focused on only melody. Dynamics, the change of loudness, adds a lot to music. In the previous sections the loudness/velocity was constant unless you changed it on the fly. We will use a very simple trick to add some imperfection, expression, feel to our music. Since we did not plan any intense or soft parts algorithmically, just adding some fluctuations to our velocity will be a great improvement. To do this we will use the **random** object again.

- Press **ctrl+shift+s** to save as your patch. Name it as **3_7**.
- Press **ctrl+e** to enter the edit mode.
- Remember the maximum velocity was 127. So let's say we want the velocity to change between 90 and 127.
- Press **ctrl+1** to create an object and put it above **makenote.** Label it **random 37.**
- Press **ctrl+1** to create an object and put it below **random 37**. Label it **+ 90.**
- Connect the output of **+ 90** to the second input of **makenote.**
- Delete the number object connected to the second input of **makenote.**
- Connect the output of the **metro** object to the input of **random 37** to have a patch like in Figure 13**.**
- Set up the metronome period, velocity and duration as usual and start the patch via the **bang**.



```
;
Cmajor 0 48 50 52 53 55 57 59 60 62 64 65 67 69 71 72 74 76
77 79 81 83 84 86 88 89 91 93 95 96
```

Figure 13: Fluctuating loudness/velocity

3.8 Duration

Again, as you might have noticed, we never bothered to have notes with different durations. To do that we will create another table/array and put some values inside it.

- Press **ctrl+shift+s** to save as your patch. Name it as **3_8**.
- Now press **ctrl+e** to enter edit mode.
- Press **ctrl+1** to create an object, and write **table Times 5**.
  - Here we are creating a **table** object. Table is a type of array in Pure Data.
  - The name of the table is T**ime.**
  - The length of the table is 5.
- Now we will fill the **table**. Press **ctrl+2** to create a message and copy paste the following into the message:

;

Times 0 500 1000 2000 1000 500

- The format of the message is as follows:
  - "**;**" (semicolon) followed by **return button** "enter"
  - **Name of the array/table**
  - The index of the array/table that you want to start writing
  - The values that you want to write.
- Thus the 0th element will be 500; the 1st will be 1000 and so on.
- Since this is a **message** you need to send it. You do not need to connect it but you need to leave the edit mode via **ctrl+e** click on the **message** and enter the edit mode again via **ctrl+e**.
- Now, the patch should look like Figure 14 if you are continuing from the previous section.
- Press **ctrl+1** to put the object above and right of **makenote** and label it **tabread.**
  - As you usual feel free to arrange the location of the objects.
- Press **ctrl+2** to create a message, put it above **tabread** and write **set Times** inside it.
- Connect the output of the **set Times** message to the input of the **tabread** object.
- Since this is a **message** you need to send it. Leave the edit mode via **ctrl+e** click on the **message (set Times)** and enter the edit mode again via **ctrl+e**.
- Press **ctrl+1** to put the object above of **tabread** and label it **random 5.**
- Connect the output of the **metro** to the input of **random 5.**
- Connect the output of the **random 5** to the input of **tabread.**
- Delete the **number** object that is the connected to the third input of **makenote.**
- Connect the output of the **tabread** to rightmost input of **makenote** as in Figure 15**.**

- Set up the metronome period, velocity and duration as usual and start the patch via the **bang**.

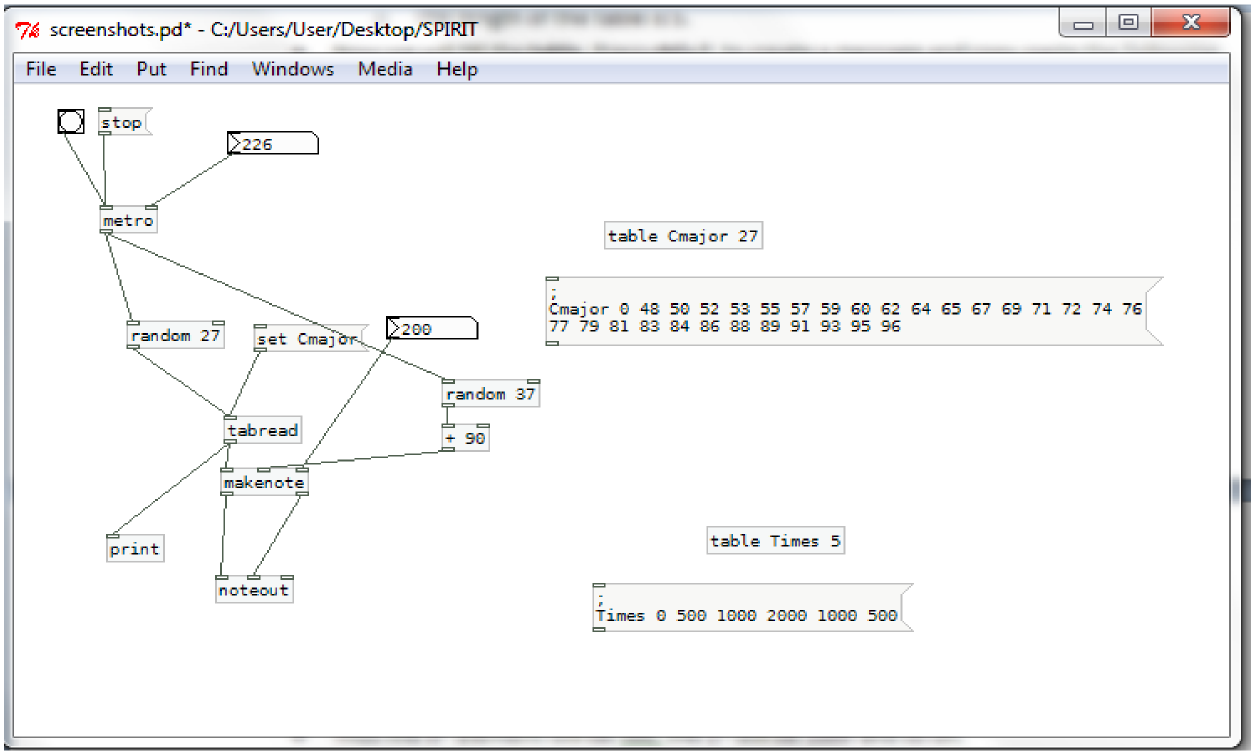

screenshots.pd* - C:/Users/User/Desktop/SPIRIT
File Edit Put Find Windows Media Help
stop
226
metro
table Cmajor 27
; Cmajor 0 48 50 52 53 55 57 59 60 62 64 65 67 69 71 72 74 76 77 79 81 83 84 86 88 89 91 93 95 96
random 27
set Cmajor
200
random 37
+ 90
tabread
makenote
print
noteout
table Times 5
; Times 0 500 1000 2000 1000 500

Figure 14: Times table

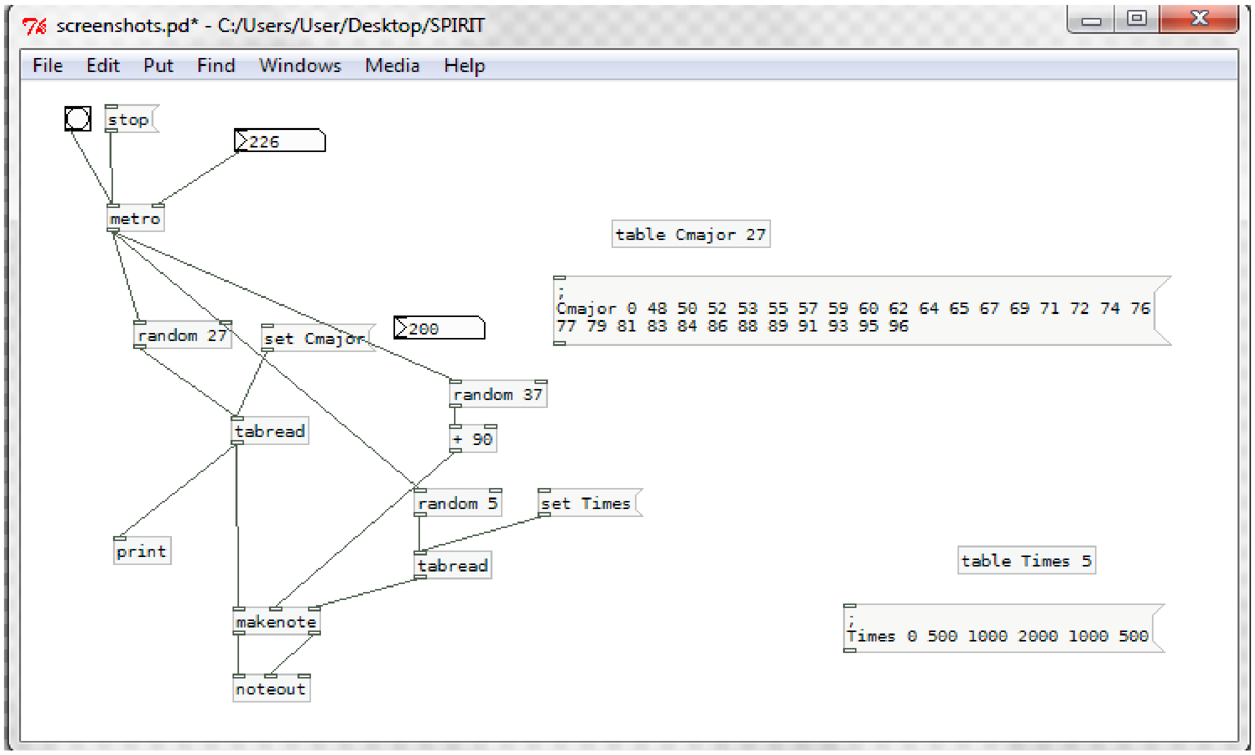

screenshots.pd* - C:/Users/User/Desktop/SPIRIT
File Edit Put Find Windows Media Help
stop
226
metro
table Cmajor 27
; Cmajor 0 48 50 52 53 55 57 59 60 62 64 65 67 69 71 72 74 76 77 79 81 83 84 86 88 89 91 93 95 96
random 27
set Cmajor
200
random 37
+ 90
tabread
random 5
set Times
print
tabread
table Times 5
; Times 0 500 1000 2000 1000 500
makenote
noteout

Figure 15: Fluctuating duration

3.9 Polyphony

The next thing that we will add in this tutorial is introducing polyphony. We will have another instrument, albeit in a very simple way. If we limit both instruments to play in the same key, they will not sound great but they will not sound bad.

- Press **ctrl+shift+s** to save as your patch. Name it as **3_9**.
- Let's say we will add a bass line. We need to create a **table** with lower notes.
- Now press **ctrl+e** to enter edit mode.
- Press **ctrl+1** to create an object, and write **table CmajorBass 22**.
    - Here we are creating a **table** object. Table is a type of array in Pure Data.
    - The name of the table is **CmajorBass.**
    - The length of the table is 22.
- Now we will fill the **table**. Press **ctrl+2** to create a message and copy paste the following into the message:

;

CmajorBass 0 36 38 40 41 43 45 47 48 50 52 53 55 57 59 60

- The format of the message is as follows:
    - "**;**" (semicolon) followed by **return button** "enter"
    - **Name of the array/table**
    - The index of the array/table that you want to start writing
    - The values that you want to write.
- Thus the 0th element will be 36, the 1st will be 38 and so on.
- Since this is a **message** you need to send it. You do not need to connect it but you need to leave the edit mode via **ctrl+e** click on the **message** and enter the edit mode again via **ctrl+e**.
- Move these somewhere empty.
- Press **ctrl+1**, put the object somewhere and label it **drunk.**
- Press **ctrl+2** to create the message for **drunk.** Put it above **drunk** and write "11 22 4" inside.
- Connect the **message** to the **drunk** object.
- Leave the edit mode, send the message and enter the edit mode.
- Press **ctrl+1,** put the object under **drunk** and label it **tabread.**
- Press **ctrl+2** to create a message, put it above **tabread** and write **set CmajorBass** inside.
- Connect the output of the **set CmajorBass** message to the input of the **tabread** object.
- Since this is a **message** you need to send it. Leave the edit mode via **ctrl+e** click on the **message (set CmajorBass)** and enter the edit mode again via **ctrl+e**.
- Connect the output of the **drunk** object to the input of the **tabread** object.
- Press **ctrl+1,** put the object below and label it **makenote.**

- Press **ctrl+1,** put the object below and label it **noteout 2.**
  - This will send the output to the 2nd channel.
- Connect the output of the **drunk** to the input of **tabread.**
- Connect the output of **tabread** to the first input of **makenote.**
- Connect the first output of **makenote** to the first input of **noteout 2** and connect the second output of **makenote** to the second input of **noteout 2.**
- Connect the output of **metro** to the input of drunk.
  - By this time probably you need a very long wire. So we will use something different.
- Delete the wire you just connected.
- Press **ctrl+1,** put the object just below **metro,** and label it **send m.**
  - Here, we created a **send** object with the identifier **m.**
  - **Send** object works with **receive** object to have a wireless connection between Pure Data objects.
  - **Send** and **receive** objects are linked through their identifiers.
- Connect the output of **metro** to input of **send m.**
- Press **ctrl+1,** put the object just below **+ 90** of fluctuating velocity**,** and label it **send v.**
- Connect the output of **+ 90** to the input of **send v.**
- Press **ctrl+1,** put the object just below **tabread** of fluctuating duration**,** and label it **send t.**
- Connect the output of **tabread** to the input of **send t.**
- Press **ctrl+1,** put the object just above **drunk** of the 2nd instrument (the one we added in 3.8)**,** and label it **receive m.**
- Connect the output of **receive m** to the input of **drunk.**
- Press **ctrl+1,** put the object just above **makenote** of the 2nd instrument**,** and label it **receive v.**
- Connect the output of **receive v** to the second input of **makenote.**
- Press **ctrl+1,** put the object just above **makenote** of the 2nd instrument**,** and label it **receive t.**
- Connect the output of **receive t** to the third input of **makenote.**
- Now it should look like Figure 15.

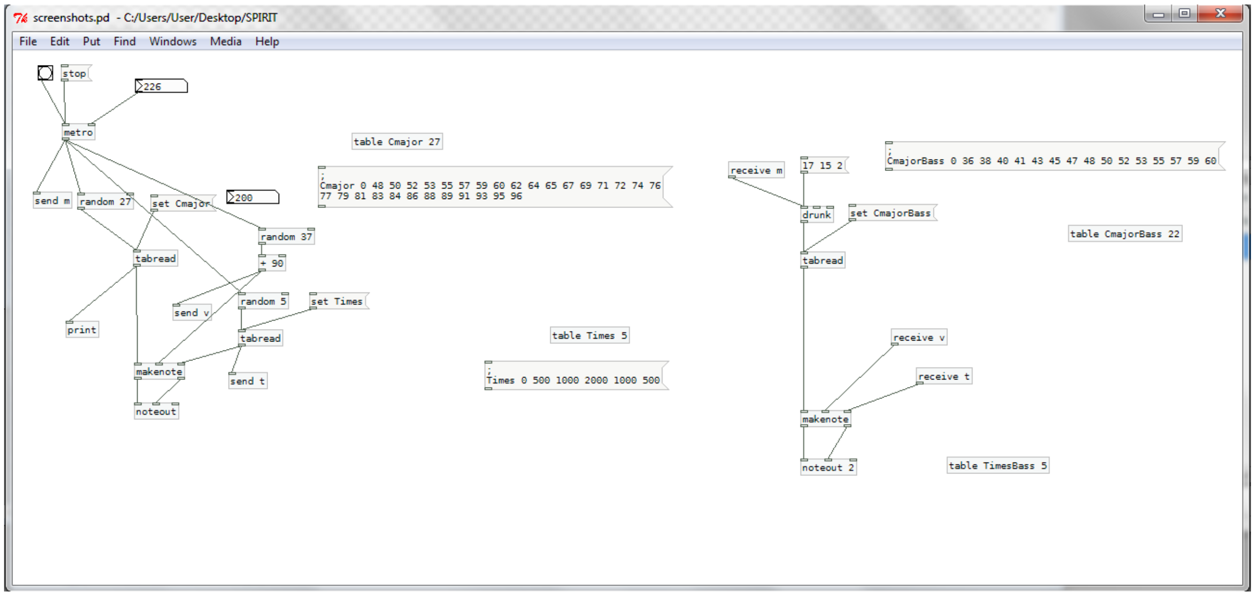

screenshots.pd - C:/Users/User/Desktop/SPIRIT
File Edit Put Find Windows Media Help
stop
226
metro
send m
random 27
set Cmajor
200
random 37
+ 90
tabread
print
send v
random 5
set Times
tabread
makenote
send t
noteout
table Cmajor 27
;
Cmajor 0 48 50 52 53 55 57 59 60 62 64 65 67 69 71 72 74 76 77 79 81 83 84 86 88 89 91 93 95 96
table Times 5
;
Times 0 500 1000 2000 1000 500
receive m
17 15 2
drunk
set CmajorBass
tabread
;
CmajorBass 0 36 38 40 41 43 45 47 48 50 52 53 55 57 59 60
table CmajorBass 22
receive v
receive t
makenote
noteout 2
table TimesBass 5

Figure 15: Added polyphony

4) Discussions

With the last patch we are able to create a two-instrument music that is in C major scale with some dynamics and rhythm. As you can guess, there still unlimited amount of changes we can make. However, first of all you must not forget that experimentation is an important aspect here and there is no right or wrong. Below are some suggestions and ideas for further changes.

4.1 Indeterminacy

So far we have never used a deterministic approach, which means whenever we clicked our bangs we get something different. You can use some rules in addition to random numbers to achieve a different style. One basic concept in music theory is leading tone, where the last tone of a scale needs to be followed by the first tone of the scale.

To whom that are interested in music, Pure Data or algorithmic composition; I advise you to try to implement that capability. However, you can also find an implementation of it in file package.

4.2 Timbre

Timbre is defined as the property of sound that causes the difference between the same notes, in same loudness played from two different instruments (say a guitar and a piano). In our patches we did not have change in timbres (no sound effects kicking in) and our second instrument was also again a MIDI piano, but do not forget that timbre is also another variable you can program and create an algorithm for.

4.3 Silence

Silence is generally much underrated in today’s music with the record companies trying to make their music as loud as possible[3]. However, good usage of silence or empty space adds to music and makes the sound more valuable.

4.4 Dissonance

Our music was in the end very consonant: everything was in C major scale. Dissonance does not mean ugly or bad music. Again good usage of dissonance improves the music while the pure random music sounds cacophonic. You may experiment with giving the instrument different scales and try to find you own sweet spot about the amount of common notes in two scales.

4.5 Imperfection

If you are after computer perfection, feel free to keep everything synchronized perfectly. However, humans are used small degrees timing errors and changes. For example, you can introduce small amount of random errors to durations, metro period. This is especially more important in a piece with multiple instruments as those tiny timing variations open up windows, where you can hear an instrument clearer. Imagine that there is a string quartet and all of them are hitting notes at the same time, to most people it will sound like a just one big sound. But if they are not perfectly synchronized (but still synchronized) you can hear the individual tones easier.

4.6 Rhythmic and Musical Structure

In our Times array we had 500, 1000, 2000, 1000 and 500 ms. This was giving us a weighted distribution with 500 and 1000 having more chance to be randomly chosen. Yet, our piece did not have any structure even a simple one such as a bar. With structure it will be easy to direct the music to a point you like.

4.7 Progression

Again, progress was another element that our music lacked. A simple way to do a progress is to defining starting and ending points, and after that filling in between. This might be a chord progression or a melodic progression but it will give music some resolution and more meaningful tension.

[3] http://www.npr.org/templates/story/story.php?storyId=122114058

5) Files

The files that accompany this tutorial can be matched with the chapters of the tutorial as follows:

File Name	Chapter No / Explanation
MIDI_basics.pd	3.2
Random_1.pd	3.3
Random_2.pd	3.4
Drunk_1.pd	Drunk without frequency limit
Drunk_2.pd	3.5
Scale_1.pd	3.6 Random within C major
Scale_2.pd	3.6 Drunk within C major
Dynamics.pd	3.7
Duration.pd	3.8
Polyphony.pd	3.9
Determinacy.pd	4.1
MasterPatch.pd	Includes all above and more in a single patch