Bytebeat and RPN Guide

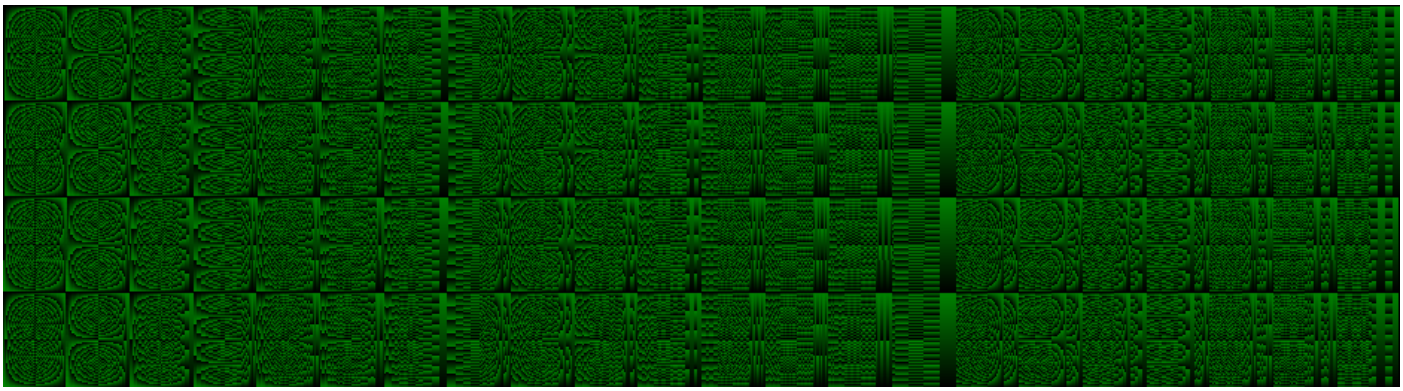


Image: Fever dream by ravary

Hello, I'm ravary and in this document I would like to explain in my own way some things about bytebeat and the RPN syntaxis, I truly recomend reading the espectacular TuesdayNightMachines absolute beginner's guide to coding Bytebeats. This guide is for anyone interested in learning bytebeat in a different way, learn some new bytebeat tricks or learn to use the RPN syntaxis, RPN syntaxis is escential in creting music for Madgarden's game EGGNOGG+, wich have bytebeat RPN music in it. My hope is that some day, a group of experts get the motivation to fully understand bytebeat and make it more simple to learn and to use, and make a better guide than I did.

like a lot of people I'm not a music expert or an advance mathematician, but Im very interested in Bytebeat, I might not be an expert, but is pretty hard to become an expert in bytebeat, because there is a lot of information but not a lot of answers, as (Jeffrey Morris,2019) stated:

"it may be disheartening at first to recognize that you may never "master" it—that it is at the same time deterministic and also undeterminable"

INDEX

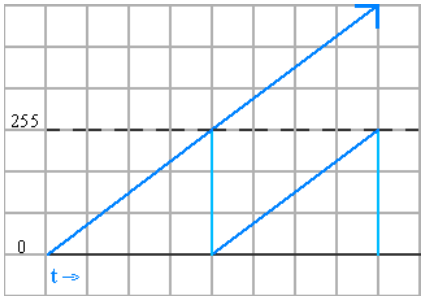
What is bytebeat?

Bytebeat is a Low-complexity art type of 8bit music originally written for small c programs by Ville-Matias Heikkilä (also known as Viznut) ussualy using Javascript expressions and bitwise logic, but can also use RPN (reverse polish notation) insted of the usual infix syntaxis, as seen in Madgarden Glitch Machine; bytebeat music is the result of alterating a sawthooth wave variable “t” (time) with formulas producing an 8 bit output at usually 8kHz, it is called bytebeat because 8bits equal to 1 byte and this byte music beats.

What is “t”?

“t generates a sawtooth wave with a cycle length of 256 bytes resulting in a frequency of 31.25 Hz when using the the default sample rate of 8000 Hz” (Heikkilä,2011)

“t” isn’t exactly a sawthoot wave itself, in reality “t” is an infinite diagonal line that it is wrapped around an 8 bit output (256 numbers) wich is the amplitud of the sawthoot wave formed. And when it reaches 255 “t” goes back to zero repeating itself.



How to bytebeat?

First we need a program to play bytebeat, we will be using the greggman HTML5 Bytebeat player in this guide.

In the program you will see our good old “t”, now to play some music we need to modify it by making some “code”.

How to “code” bytebeat?

We wont be exactly “coding”, this is just how some people call the process of making a bytebeat song, so lets start with the:

Arithmetic operators
+ Sum

if you try to sum normal numbers to “t” it wont drasticly change, but if we sum it by itself “t + t” it will produce a higher pitch sound, but what is this sum if not “2 * t”.

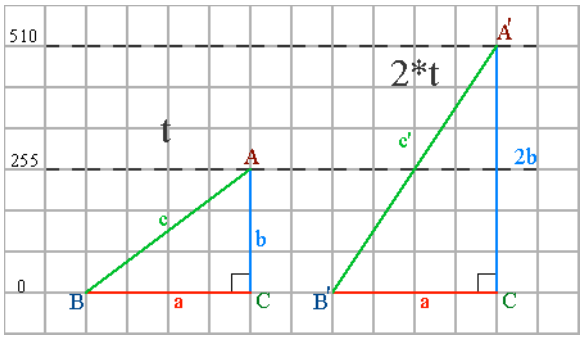
*** Multiplication**

is just multipliying the speed of time (t), so “ 2 * t ” will just be 2 times faster than “t”, and “ 2 * t ” is the same as “ t * 2 ”.

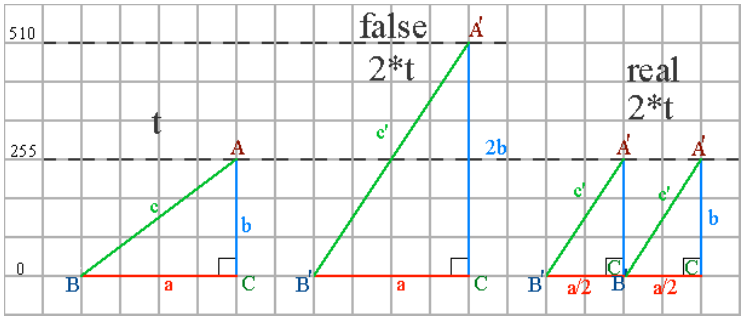
Like an audio played at two times the speed, multiplying “t” will make the pitch higher. As (Heikkilä,2011) said:

“The pitch can be changed with multiplication, t*2 is an octave higher, t*3 goes up by 7 semitones from there…”

the way multiplication seems to functions like this;



If we multiply “t” by two (2*t), we start by multiply the “b” height: 255*2=510 ; as we see “b” has double its size, hypotenuse “c” has grown in size becoming “ c’ ” and as the inside angle of “A” decreases becoming “ A’ ”, the inside angle of “B” increases becoming “ B’ ”.



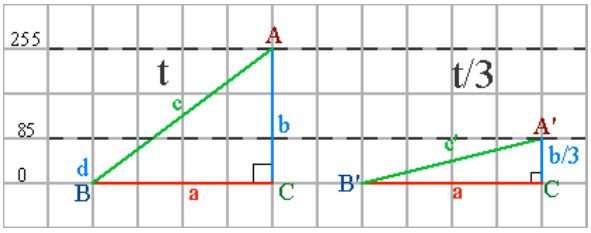
But “t” is wrapped around 255, so now the real“2*t” will have the same height and base as the original “t” , but the same angles as the false “2*t”.

/ Division

is just dividing the speed of time (t), so “ t / 2 ” will just be 2 times slower than “t”. but we need to be carefull not to think that “ 2 / t ” and “ t / 2 ” are the same, because different than the multiplication, the order in the division is important and do affect.

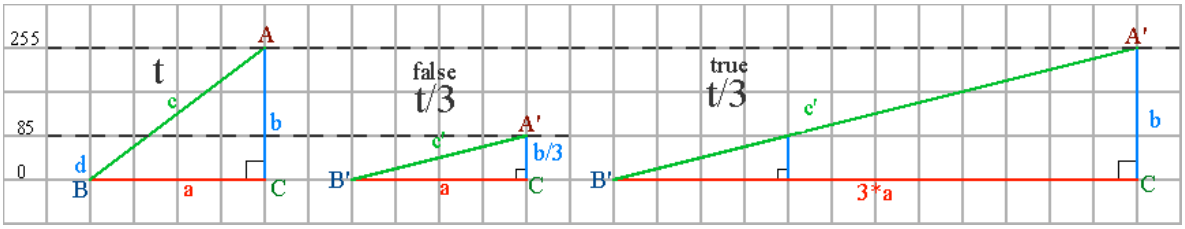
As we stated before multiplication will make the the ptich higher and “t*2” is going an octave higher, continuing this, the division will make the pitch lower, so “t/2” will go an octave lower and “t/3” goes down by 7 semitones from there.

the way division functions is like this;



if we wanted to divide “t” by three (t/3), we start by dividing the “b” height: 255/3=85; as we see “b” is now a third of what it was, the hypotenuse “c” has shorten its size becoming “ c’ ” and as the inside angle of “A” increases becoming “ A’ ”, the inside angle of “B” decreases becoming “ B’ ”.

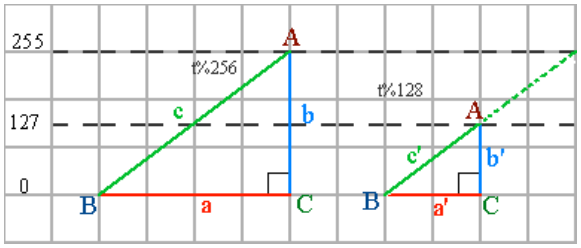
But we need to remember that “t” is an infinite diagonal line that is wrapped around and output of 255, so the hypotenuse “ c’ ” will continue until it reaches the 255 output becoming the real “t/3”, while maintaining the same angles as the false one.



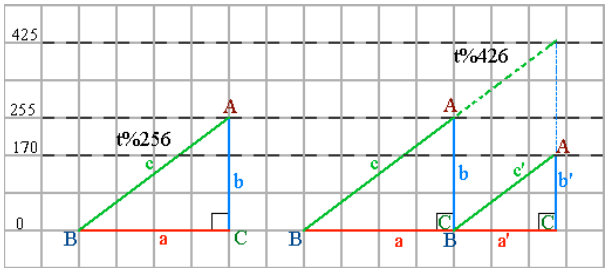
% Modulo

the modulo wraps the amplitud (volume) of “t” by the number given, but the max output will stay the same (8bit output=256).

so if we wrapped “t” by 256 (t%256) it will stay the same as “t” with an output from 0 to 255, but if we wrapped “t” by 128 (t%128) it will have an output from 0 to 127 while maintaining the same angles as “t%256”

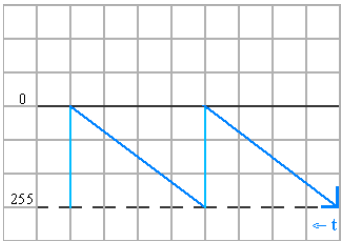


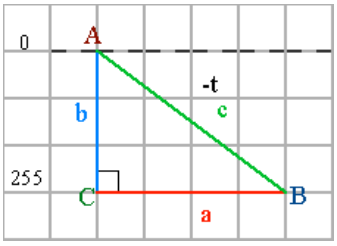
But what happens if we wrapped “t” by a number greater than 256, well “t”will be wrapped by 256 and instantly be wrapped by the rest of the number given. In this example “t” is wrapped by 426, so “t” will go on until it is wrapped by the 256 output and then instantly will repete itself until it goes to the remaining output of 426 that its 171 (425-255=170).



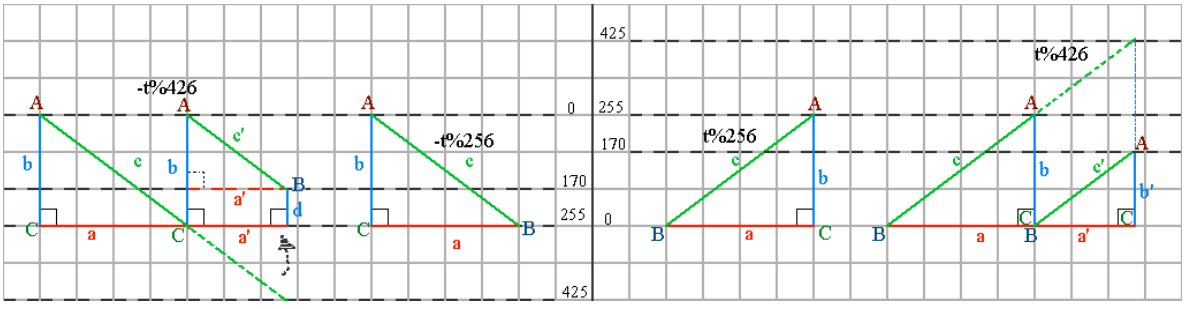
-t (minus t)

But what happens if we invert “t” (-t), well it will be flipped in a strange way, now when you want to wrap it the metod is different, (Im not sure if this is true, but it seems to work) because “-t” is now heading in the oposite direction as “t” and “-t” still starts at 0 and goes until it is wrapped on 255, but now 255 is down and 0 is up



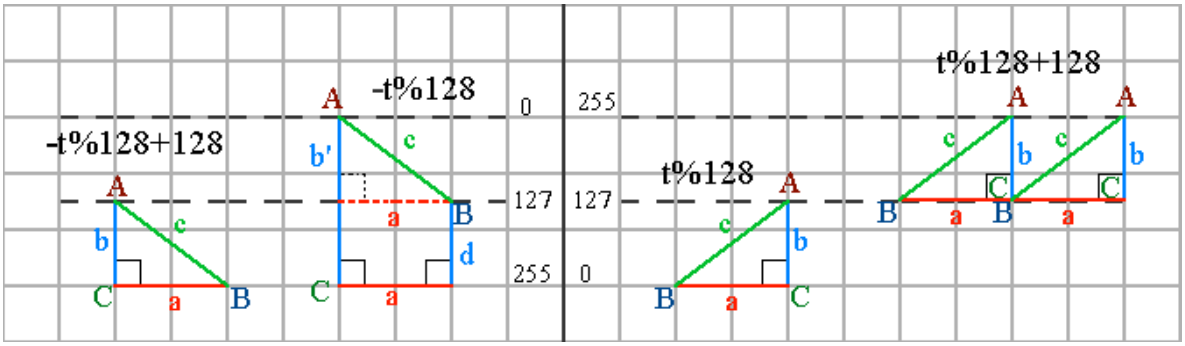


Lets see this comparasion between “-t%426” and “t%426”



If you understand the image you can se that in the reverse t, a new height is made “d” wich is just the remaining of 255-170 wich is 85. This height “d” isnt relly a height itself but the “b” height of another sawthoot wave at its side.

Now, taking the TuesdayNightMachines example “t%128+128” in this expresion “t” isn’t going from 0 to 127, now is goint from 128 to 255; but what will happen if we invert “t” and make the expresion “-t%128+128”, well it will be almost the same, because insted of going from 0 to 127 is now going from 128 to 255.



Bitwise operators

A bitwise operation operates on a bit string at the individual bits level,this topic is more complex, so if you want to understand more about the Bitwise operators, I reccomend watching Back To Back SWE video about it. In bytbeat this operators function as audio mixers in different ways , except the bitshift, that one works different.

& and

This operator can give alternating values as in “t&128”, wich will output only values from 0 to 128. But you can do more things, like miixing “t&t/20” this sound will give a rising sound.

| or

This operator is the most used as an audio mixer, compare the sound of “t&t/20” and “t|t/20”, the “or” one will sound more like a videogame tune.

^ xor (exclusive or)

This operator will give more chaotic or distorted sounds, just compare the sound of “t|t/20” and “t^t/20”

>> Right bitshift

works like a division, take a look at “t>>2” and “t/4”, they are pretty much the same because if we subtract them they seem to cancel “(t>>2)-(t/4)”, they are the same because in the right bitshift “t” is divided by an exponent of 2. So if you want to determine the bitshift equivalent to a division is as simple as using this logarithm “$\log_2(x)$” where the argument “x” is the divisor of “t”, if I wanted the bitshit equivalent of “t/64” is as simple as “$\log_2(64) = 6$” so “t>>6” will be our equivalent, and in the other way around, if we wanted the division equivalent of a bitshift the formula is “2^x”

where the exponent “x” is the number of the bitshift, and the result will be our divisor, so “$2^6 = 64$”.

The fact that bitshift is pretty much like a division doesn’t mean that they will produce the exact same sound in a code, for example, if you compared “t*(t>>6)” and “t*(t/64)”, the bitshifted one will produce a more robust sound than the divition thay sounds like a soft tv tuning. This happens because the bitshift gets process faster by the computer than a division thus producing a slightly different sound.

And if you didn’t assume this already, as I said before that “t*2” will increase “t” by an octave and “t/2” will make “t” decrease by an octave, the right bitshift will decrease “t” by octaves.

<< left bitshift

works like a multiplication, in the right bitshift “t” is multiplied by 2 with an exponent, but the left bitshift doesn’t work in the same way as right bitshift works like a division, because if you left bitshift “t” from 0 to 8 you will stop hearing sound at “t<<8” and you wont be hearing sound again until “t<<32”, but the sound will stop in the same way at “t<<40”, so the sound only hears 8 times and again you wont hear any sound until “t<<64” and you wont hear anything at “t<<72”. If you havent found a pattern yet let me tell you, in the left bitshift, you can only hear a sound at intervals of 8 every mulitple of 32 (including 32*0, wich give us the first 8 sounds t<<0 through t<<7). Knowing this doesn’t matter a lot, because “t<<32” will produce the exact same sound as “t<<64” not like in the right bitshift where the division is slightly different every time so it doesn’t matter if you multiply it by “t” like in the right bitshift.

If you wanted to know the exact bitshift equivalent to a multiplication the formula is “2^x” where the x is the number of the bitshift 0 through 7, and that number need to be multiplied by “t”. We can see they are the same because they cancell every time (t<<0) - (t), (t<<3) - (8*t), all of these produce no sound.

relational operators

the relational operators as the name suggest compare and tests some kind of relation between two values and return a true or false

== equal to

This can be use to play a single sound depending on the time, example:
“(t==16000)*t ” in this code when the timer reaches 16000, it will play the sound “t”, but only for that exact time(at 8kHz rate, it will take 2 seconds), so this is only usefull to make some random noises at lower Frequencys

!= not equal to
This is the opposite of equal to, it can be use to stop the code in a exact time, example: “(t!=8000)*t ” this code will play the sound “t” except for when the timer reaches 8000, in that exact instant it stops and continues playing afer 8000.

> greater than
This is the one may use the most, because it will play the code after the time you write, example:“(t>16000)*(t>>t)” the code “(t>>t)” will play after the timer reaches 1600.

< less than
it will play the code before the time you write, example:“(t<16000)*(t<<t)” the code “(t<<t)” will play since the start until the timer reaches 1600.

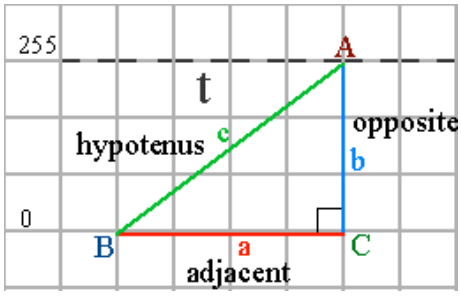
>= greater than or equal to
will work almost the same as greater than

<= less than or equal to
will work almost the same as less than

combined
With this, you can combine various codes using the bitwise operator | (or), for example: “(t<=16000)*(t<<t) | (t>=16000)*(t>>t)” in this code “(t<<t)” will play from the start until the timer reaches 16000 (2 seconds at 8kHz rate) and inmediatly will star playing “(t>>t)” until the end.

java Math.
you can use the Java Math Static methods (listed here) in newer bytebeat interpreters; such as Math.floor(), Math.sin() ,Math.cos(), Math.tan() Math.sqrt(), and others. You can shorten the functions by only writing the operation without the “Math.” Part.

To understand a litlle bit how to use trigonometric functions like sin, cos, tan we need to understand how they work, “t” will work as the hypotenus of our triangle.so in the function “ t*sin(t) ”, “t” is multiplied by sin of “t”, and this is the result of the opposite side (height) divided by the hypotenus “t” . so the result will give us a caothic noise sound.



Some other examples:

“t>>(t*tan(t))” this code make a crazy noise sound

“(t>>(t*tan(t))-t>>(t*cos(t)))” And this example sounds like a fire place

RPN

RPN stands for Reverse Polish notation, but can also be called postfix notation, I reccomend watching Back To Back SWE video about the topic so you can understand it better.

Usually you will write code in a infix notation, so lets convert an infix code to an RPN code, this process is very difficult and it may not turn out correct the first time you try it, the bigger the code and the more parentesis it has, the more difficult it will be to convert.

Lets take a look at my own song, Fever Dream by me (ravary)

t * (((t/10) | t*6)>>10)%64

the first thing to do is convert it from the inner parenthesis to the outside ones, so the first parenthesis is " t / 10 " and in the RPN syntax the simbols go at last so it will be " t 10 / " (I recommend reading it in RPN to understand it; t and 10 will be dived), the next one is " | t * 6 " so in RPN it will be "t 6 * |", the next parenthesis is ">>10" wich is "10 >>" and outside the parenthesis is "t *" wich be the same " t*" and "%64" wich will be "64 %"

so we organize them in order from the inner parenthesis to the outside ones from left to right and we get:

t 10 / t 6 * | 10 >> t * 64 %

Wich is the RPN, as we can se the RPN has less digits than the normal operation because it lacks the parenthesis, so in theory a computer will be able to process RPN notations faster than the normal infix notation.

Bytebeat coding suggestions

At the end of the day, the best way to make bytebeat music is playing with the numbers and operators, and when you find a good sound save it in a document and later try to combine it with other sound. But if you are lazy you can use the Random ByteBeat Generator by QiTaNo until you get something good to save.

- First I suggest that You use your pc speakers and not your headphones, because the headphones can make you have headaches or damage your ears.
- Sometimes in greggman html5 bytebeat player, the code wont start or will play the last code insted of the inputed code, to fix this I recomend refreashing the page with the "F5" key, and input the new code.

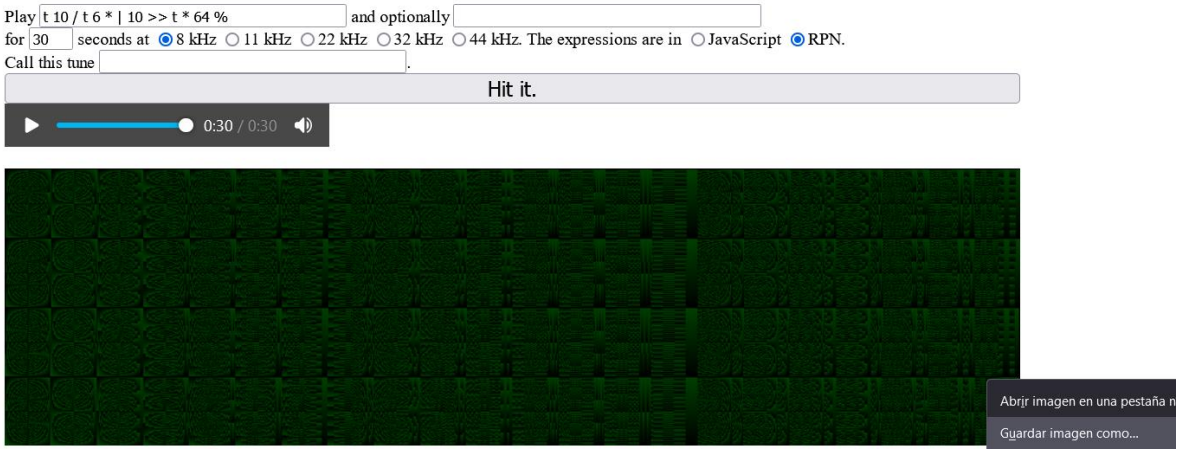
- Use variables, insted of modifying each number in your code, change the numbers to variables and modify each variable (remember to separate lines with comas)

```
a=10,
b=6,
c=10,
d=64,

t  * (((t/a)|t*b)>>c)%d
```

- Write comments to tell what each part do, like:

```
a=10, // rising sound
b=6, // high pitch sound
c=10, // shifter
d=64, // amplitud controler

t  * (((t/a)|t*b)>>c)%d
```

- Be carefull and use parenthesis, because “t>> 10 * t” is the same as “t>> (10 * t) ” and not the intended “(t>> 10) * t”
- If your code sounds slow try playing it at 11kHz or other speeds.
- If you wanted an image of your bytebeat song, I recommend you using the Darius Bacon bytebeat interpreter, use the start or the end of the song, to save the image of your song and later on you can modify the hue values on photoshop or other program to give it better colors.

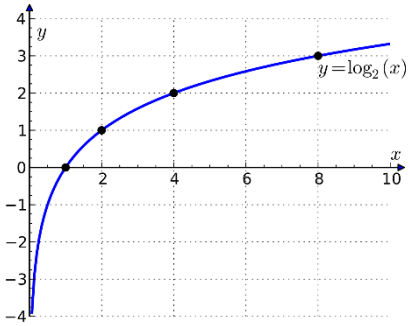


- And at the end I suggest you share your music on the reddit group r/bytebeat

What is left?

- A better explanation on music notes in bytebeat.
- A infix to RPN converter and vice versa.
- Does “-t” work the way I said?
- how does “t%128 + t” works? Why is it so chaotic?
- why the factors of 32 hearable on the left bitshift. It has to do with t not hearing when multiplying it by 256?

- the left bitshift doesn't produce sound after 7, like the right bitshift does because I think it has to do with the fact that rigth bitshift is like a division and can be converted with a logarithmic function base 2 and if we see the graph of the logarithmic function base 2, we can see that the right bitshift works at the positive "y" axis, and continues to infinity separating itself from the "x" and "y" axis; and the left bitshift works in the negative "y" axis and continues to get arbitrarily close to the"y" axis but it seems to get close at 4, but that doesnt solve the question of why it stops after 7.But I need a real mathematician to tell the thruth.



- how can you make a bytebeat bitmap images like this.
- How can people make complicated songs like Rick Astley - Never Gonna Give You Up song link, or the fornite dance song by Raphael Goulart link ?
- It will be cool to see bytebeat in minecraft, minecraft is written on java, and bytebeat has been written in java, so it doesn't sound complicated to make a mod or a command block to play bytebeat codes.

Cites

1. Ville-Matias Heikkilä . (10/ 02/2011)Algorithmic symphonies from one line of code -- how and why?, at: https://countercomplex.blogspot.com/2011/10/algorithmic-symphonies-from-one-line-of.html
2. Jeffrey Morris. (20/09/2019). Bytebeat: Deterministic and Undeterminable, at: https://www.researchcatalogue.net/view/921059/922509#ref-liljedahl
3. Felix.The Tuesday Night Machines (04/01/2020) The absolute beginner's guide to coding Bytebeat, at https://github.com/TuesdayNightMachines/Bytebeats/blob/master/Bytebeats_Beginners_Guide_TTNM_v1-5.pdf
4. Back To Back SWE (23/02/2019). Add Two Numbers Without The "+" Sign (Bit Shifting Basics), at https://youtu.be/qq64FrA2UXQ
5. Back To Back SWE (25/02/2019). Reverse Polish Notation: Types of Mathematical Notations & Using A Stack To Solve RPN Expressions, at: https://www.youtube.com/watch?v=qN8LPIcY6K4&t=460s

Further reading

- Ville-Matias Heikkila blogs
  - https://countercomplex.blogspot.com/2011/10/algorithmic-symphonies-from-one-line-of.html

    - https://countercomplex.blogspot.com/2011/10/some-deep-analysis-of-one-line-music.html
- Ville-Matias Heikkila document https://arxiv.org/pdf/1112.1368.pdf
- Jeffrey Morris document http://generativeart.com/GA2019_web/28_JeffMorris-168x240.pdf
- Some references Bytebeat — Kragen (canonical.org)
- More links https://dadabots.com/bytebeats/
- Meta filter post https://www.metafilter.com/111959/Todays-formulaic-music
- bytebeat RPN twitter bot https://twitter.com/bytebeat_bot
- gabochi bytebeat, floatbeat, rampcode https://toplap.org/bytebeat-floatbeat-rampcode/
- naivesound posts https://medium.com/@naive_sound
- Xia, G., & Noel, R. Exploring the Space of ByteBeat Music: A Genetic Algorithm. https://robertonoel.com/assets/pdf/Genetic_Algorithm.pdf

Tutorials

- bytebeat
    - TuesdayNightMachines The absolute beginner's guide to coding Bytebeats! https://nightmachines.tv/downloads/Bytebeats_Beginners_Guide_TTNM_v1-5.pdf
    - Prof. Jeffrey Morris MUSC 318 class https://youtube.com/playlist?list=PLnoBrd8xpnNS14304GfujhkK7g0OysI3W
    - Greggman github https://github.com/greggman/html5bytebeat
    - Gabriel Vinazzaultimate VIDEO-GUIDE https://youtu.be/K4GsZu8kBbw
- The Glitch machine(RPN)
    - Madgarden Archive Forum www.madgarden.net • View forum - Glitch Machine (archive.org)
    - Madgarden Archive Blog https://web.archive.org/web/20181210192606/http:/madgarden.net/apps/glitch-machine/tutorial/
    - leuLlama/@yorgle (Scott) http://umlautllama.com/w2/?action=view&page=GlitchMachine

bytebeat interpreters and proyects

- Bemmu & viznut https://web.archive.org/web/20180320174229/http://www.bemmu.com/music/index.html
- Rares fluid's http://wurstcaptures.untergrund.net/music/
- Darius Bacon https://web.archive.org/web/20180322025410/http:/wry.me/bytebeat/

- Darius Bacon github https://github.com/darius/bytebeat
- Erlehmann python libglitch https://github.com/erlehmann/libglitch
- Kragen Pybeat https://github.com/kragen/pytebeat
- Abagames bytebeatbank https://github.com/abagames/bytebeatbank
- gabochi bash functions rampcode https://github.com/gabochi/rampcode
- gabochi bytebeat visual patterns in bash, bANSh https://github.com/gabochi/bANSh
- paul_hayes https://entropedia.co.uk/generative_music/#v3b64K0otKS3KUyhR0FcwNFBQUyjRMlbQViixsSnhAgA=
- naivesound Glitch https://github.com/naivesound/glitch
- Random ByteBeat Generator by QiTaNo https://www.dropbox.com/s/lo8nnmjq2q0giho/ByteBeat%20Generator%20v1%20by%20QiTaNo.zip?dl=0
- SthephanShinkufag bytebeat-composer https://github.com/SthephanShinkufag/bytebeat-composer

Bytebeat songs and forums

- A youtube recopilation with bytebeat songs https://youtube.com/playlist?list=PLnoBrd8xpnNRt1gtenqBMDtXEWB5LwwKb
- Vizunt videos and others https://web.archive.org/web/20120301055914/http:/pelulamu.net/countercomplex/music_formula_collection.txt
- Erlehmann collection https://github.com/erlehmann/algorithmic-symphonies \
- Windows 93 bytebeat https://www.windows93.net/#!bytebeat
- Greggman github examples https://github.com/greggman/html5bytebeat
- Pouet dot net https://www.pouet.net/topic.php?which=8357&page=18
- Battle of the bits https://battleofthebits.org/academy/GroupThread/11089/bytebeat+general/
- 0xa kuri bitop videos https://web.archive.org/web/20160312150403/http://0xa.kuri.mu/2011/10/09/bitop-videos/
- Inki.fi/o http://yehar.com/blog/?p=2554
- Scene dot org https://files.scene.org/browse/parties/2021/lovebyte21/bytebeat_music/
- leuLlama/@yorgle (Scott). RPN Songs http://umlautllama.com/w2/?action=view&page=GlitchMachine
- Madgarden's Glitch Machine Forum RPN Songs
    - https://web.archive.org/web/20151022075326/http:/forums.madgarden.net/viewtopic.php?f=3&t=20

    - https://web.archive.org/web/20151022035519/http:/forums.madgarden.net/viewtopic.php?f=3&t=45
    - https://web.archive.org/web/20150912092422/http:/forums.madgarden.net/viewtopic.php?f=3&t=42
- Reddit group https://www.reddit.com/r/bytebeat/
- SthephanShinkufag Discussion threads https://dollchan.net/btb/

Credits

Made in 2021, v1.0, Graphics made with aseprite

Original research that I made https://drive.google.com/file/d/1FjLrfwSEHRXCqdixNQF7fbq1So8fxovi/view?usp=sharing

Contact info

- reddit u/Ravary212
- twitter @RavaryGamer212