

 **gabochi** / **gede**

Branch: master ▾

**gede** / OneInputTutorial /

Create new file

Upload files


















Find file












History


 **gabochi** Update readme.md

Latest commit cd2221b 12 days ago

..

 <a href="#">01.png</a>	Add files via upload	13 days ago
 <a href="#">02.png</a>	Add files via upload	13 days ago
 <a href="#">03.png</a>	Add files via upload	13 days ago
 <a href="#">04.png</a>	Add files via upload	13 days ago
 <a href="#">05.png</a>	Add files via upload	13 days ago
 <a href="#">06.png</a>	Add files via upload	13 days ago
 <a href="#">07.png</a>	Add files via upload	13 days ago
 <a href="#">08.png</a>	Add files via upload	13 days ago
 <a href="#">09.png</a>	Add files via upload	13 days ago
 <a href="#">10.png</a>	Add files via upload	13 days ago
 <a href="#">11.png</a>	Add files via upload	13 days ago
 <a href="#">12.png</a>	Add files via upload	13 days ago
 <a href="#">13.png</a>	Add files via upload	13 days ago
 <a href="#">14.png</a>	Add files via upload	13 days ago
 <a href="#">15.png</a>	Add files via upload	13 days ago
 <a href="#">16.png</a>	Add files via upload	13 days ago
 <a href="#">17.png</a>	Add files via upload	13 days ago

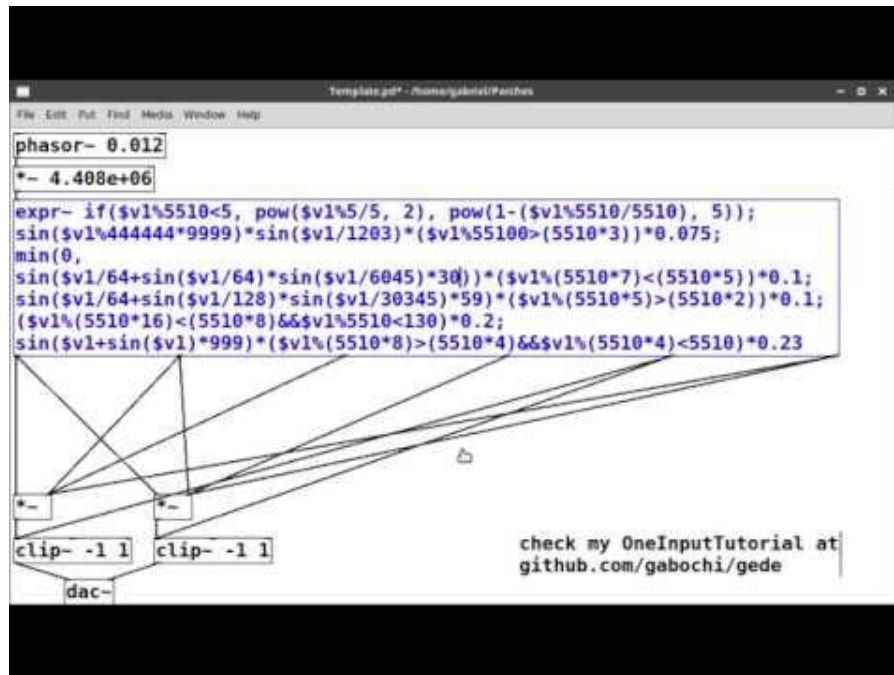
 <a href="#">18.png</a>	Add files via upload	13 days ago
 <a href="#">19.png</a>	Add files via upload	13 days ago
 <a href="#">20.png</a>	Add files via upload	13 days ago
 <a href="#">21.png</a>	Add files via upload	13 days ago
 <a href="#">22.png</a>	Add files via upload	13 days ago
 <a href="#">23.png</a>	Add files via upload	13 days ago
 <a href="#">24.png</a>	Add files via upload	13 days ago
 <a href="#">cypypaste.txt</a>	Add files via upload	13 days ago
 <a href="#">demo-2.pd</a>	Add files via upload	14 days ago
 <a href="#">demo.pd</a>	Add files via upload	14 days ago
 <a href="#">readme.md</a>	Update readme.md	12 days ago

 [readme.md](#)

# Hola, bienvenido al OneInput Tutorial

Hi, if you don't get spanish you can follow the images, see the vid and download the demo patches (all that is in eng).

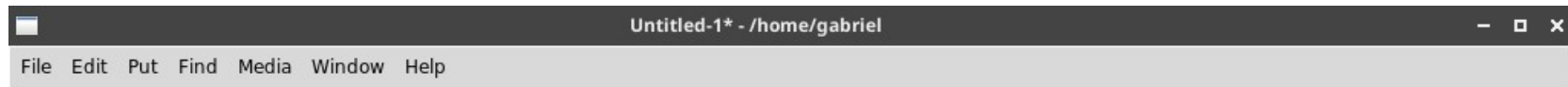
Si todavía no viste el video de demostración que está en YouTube y estás ansiosx por saber qué y cómo se puede hacer:



Me gusta cómo suena a eso de los 10:30 pero vale la pena mirarlo para darse una idea del proceso y de cómo, en una improvisación, se trabaja para darle forma a lo que va surgiendo. Ahora siguen una serie de imágenes bastante claras de cómo es la cosa.

**AL TRABAJAR CON EXPRESIONES MATEMÁTICAS Y LÓGICAS, PRÁCTICAMENTE TODO ES APLICABLE A TODO**, así es que no tomes este tutorial como si fuera lo único que puede hacerse sino como una base a partir de la cual podés empezar a experimentar por tu cuenta.

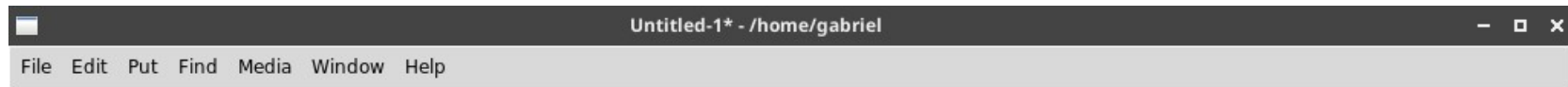
La idea es trabajar solamente con *expr~* a partir de una única señal de entrada, algo valioso sobre todo en un contexto de livecoding. En esta carpeta podés encontrar un archivo **copypaste.txt** que contiene las principales fórmulas que aparecen acá para que no tengas que tipear o recordar todo a la perfección. Si te interesa le vas a ir agarrando la onda y entendiendo mejor.



# Hi! Welcome to the OneInput Pure Data livecoding tutorial

## phasor~

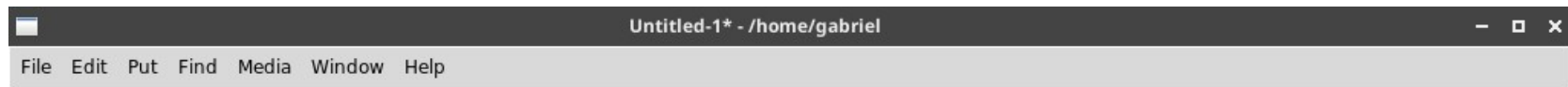
Esta rampa va a ser nuestra One(and only)Input para *expr~*. Podés cambiar los valores alterando la **velocidad, tono y duración del loop**. En este caso, el loop dura 100 segundos. Eecordá que **TODO depende de esta función**, de manera que al volver a empezar se resetean todas las modulaciones también y por eso conviene que sea largo y produzca muchas variaciones antes de recomenzar).



**phasor~ 0.01**

**\*~ 4408000**

**First make a slow ramp**



**phasor~ 0.01**

**\*~ 4.408e+06**

**expr~ Here's where the magic will happen**

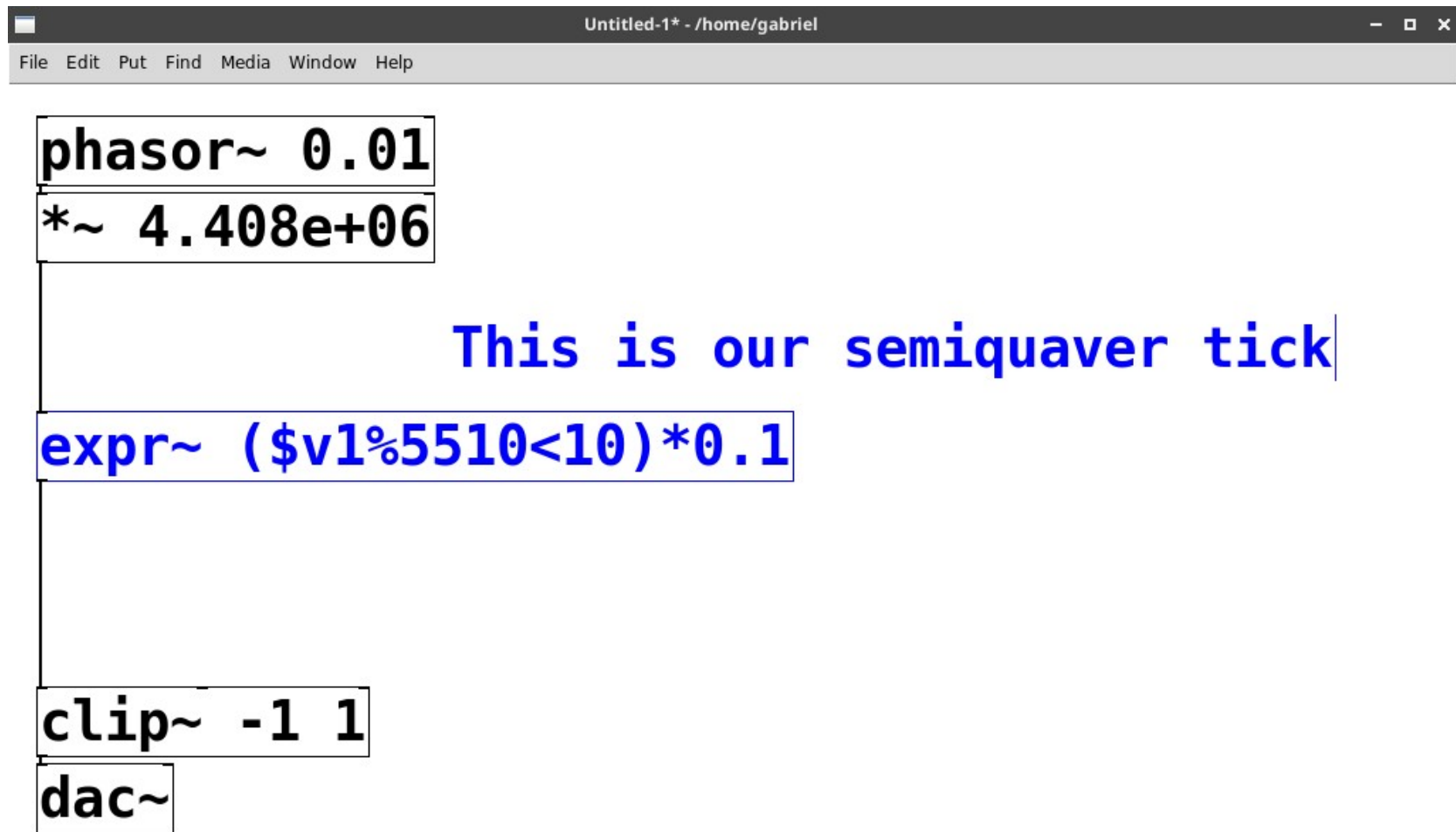
**clip~ -1 1**

**dac~**

pop

(\$v1%5510<10)

[illegible]



Si entendiste esto vas a entender todo, no hay nada que pueda impedirte terminar esta tutorial.

## Formas de onda

Veamos cómo generar distintas formas de onda operando con la señal.

**saw**

```
($v1%400/400)
```

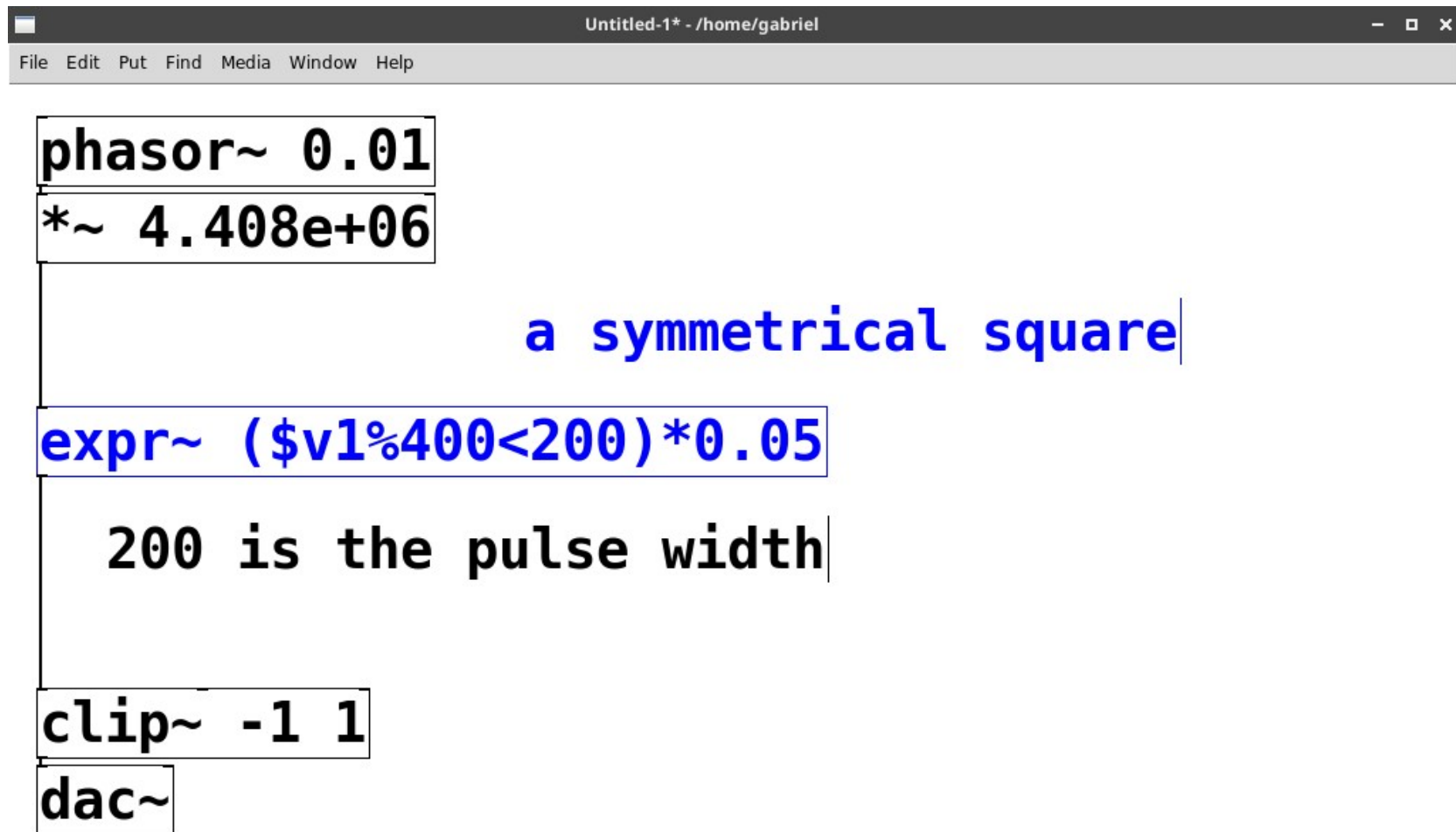


01234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901  
23456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123  
456789012345678901234567890123456789012345678901

**dac~**

(\$v1%400<200)

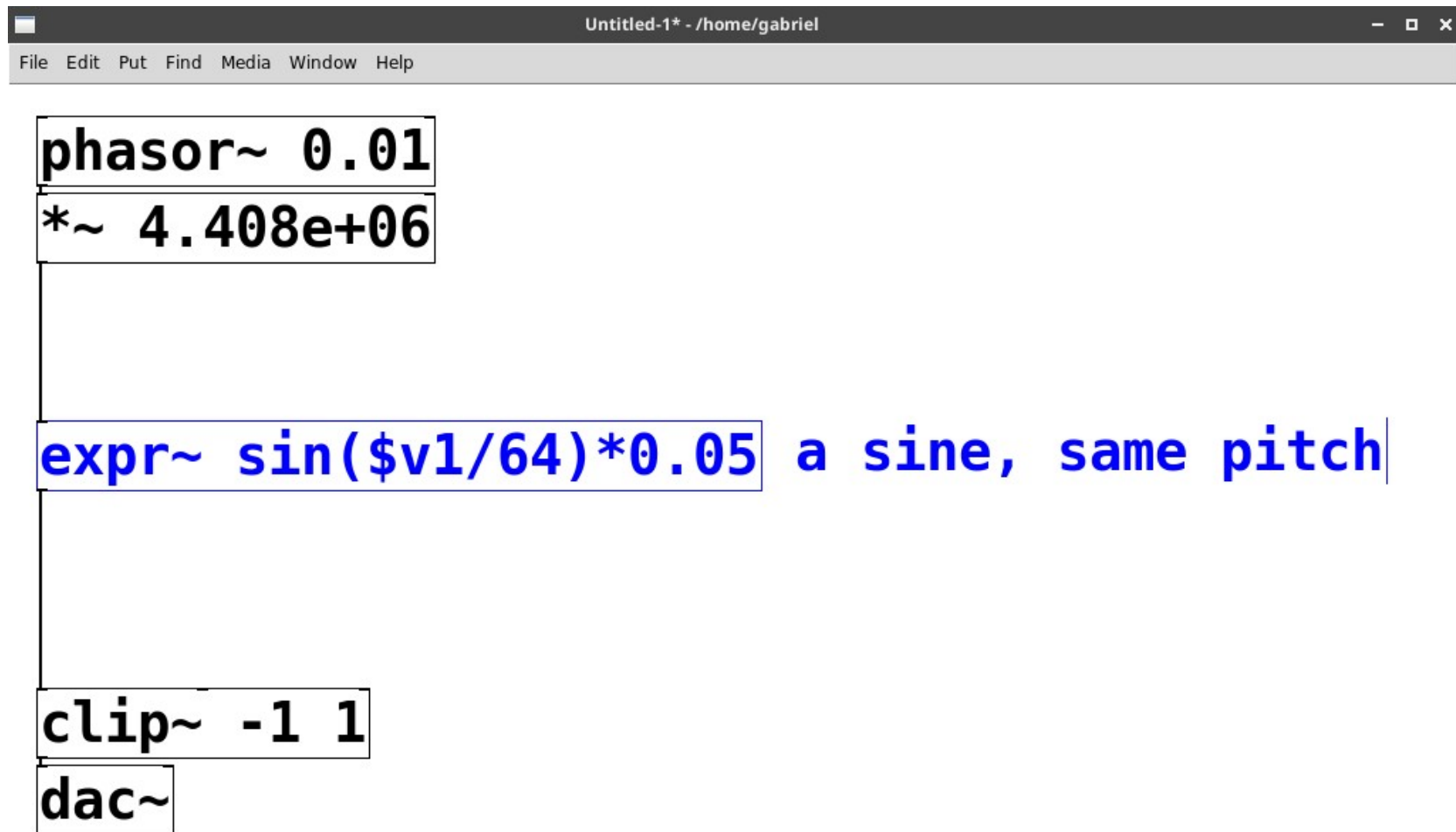
[illegible]



sine

```
sin($v1/64)
```

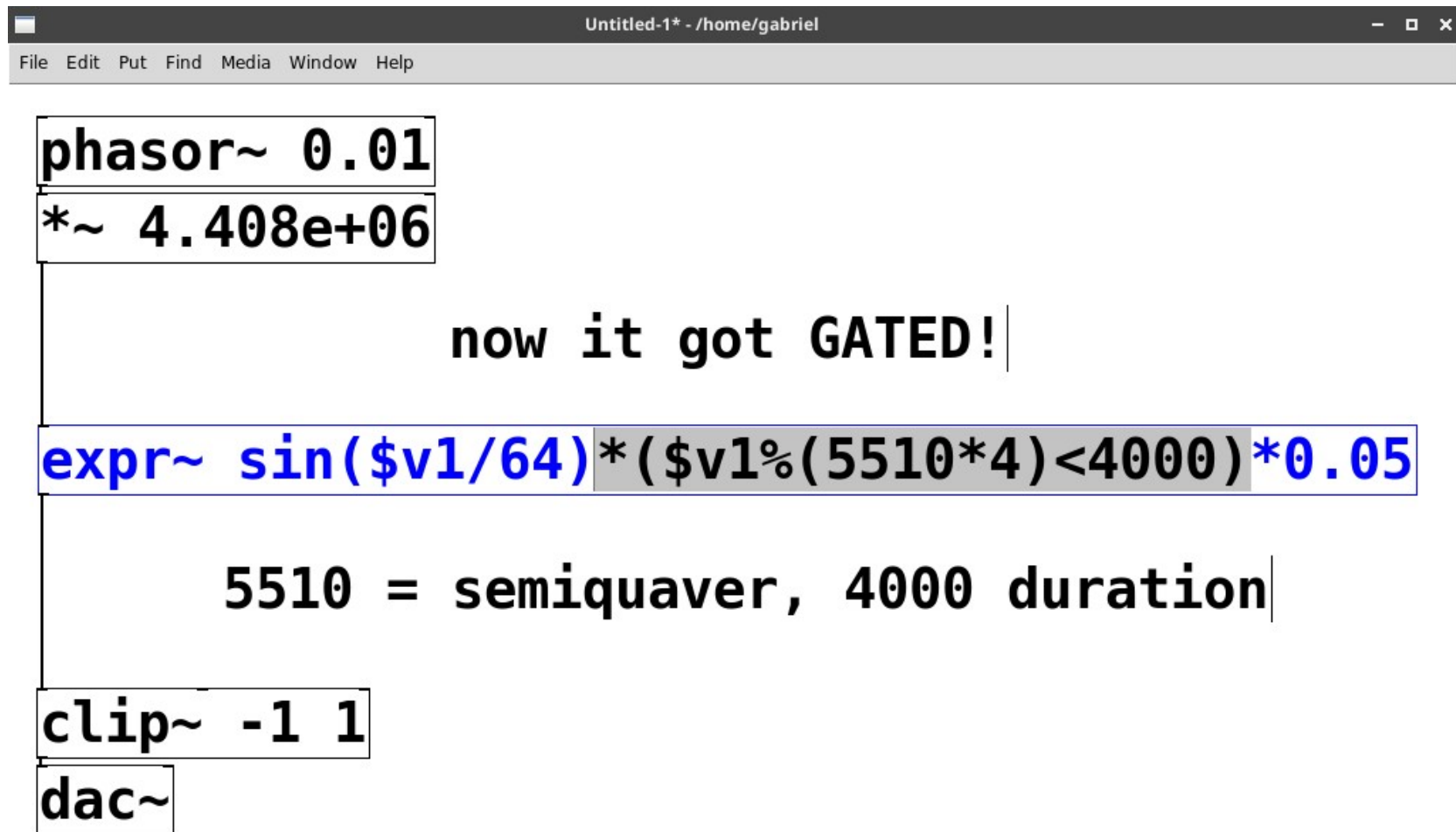
Aunque la onda sinusoidal tiene menos impacto armónico en principio, vamos a ver más adelante que resulta increíblemente valiosa. Sobre todo porque **podemos pasar cualquier cosa por seno y siempre nos va a devolver un resultado interesante entre -1 y 1.**



## GATE

```
($v1%(5510 * 4)<4000)
```

Antes de ver cómo se puede resolver el problema de los envolventes tengamos en cuenta que en ciertos contextos podemos prescindir de ellos y que, aún si vamos a usarlos, las compuertas **sirven para controlar el cuándo y el cuánto de cada sonido**. En este caso el cuándo es cada cuatro semicorcheas.



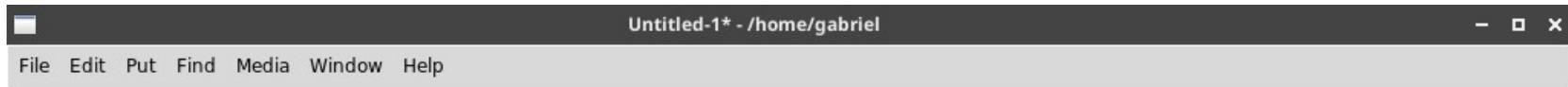
No es más que otra onda cuadrada más lenta (*a tempo*) por la que multiplicamos nuestra forma de onda. Podemos hacer lo mismo con una saw por ejemplo:

```
($v1%5510/5510)
```

De hecho eso ya es casi un envolvente decente, creo que debería utilizarlo jaja.

## Modulando

Con unos pocos elementos ya estamos en condiciones de ir al punto de la cosa: las modulaciones. Modular es todo en este contexto, el desafío es transformar esas ondas monótonas en hermosos desarrollos que nos seduzcan por sus progresiones y a la vez nos cautiven con una cierta inasible complejidad. Bueno, eso es para el libro...



**phasor~ 0.01**

**\*~ 4.408e+06**

**Let's modulate things!**

**clip~ -1 1**

**dac~**

AM

(sin(\$v1/1000) \* 0.5+0.5)

Dijimos que `sin()` retorna valores entre -1 y 1, si lo normalizamos a 0 y 1 podemos controlar la amplitud de las ondas y generar **trémolos, fades, polirritmia, etcétera**. Es, otra vez, un elemento muy sencillo pero que rinde un montón. Al principio parece una pavada pero si lo sabemos combinar con otras cosas se vuelve casi el alma, algo imprescindible.

```

phasor~ 0.01
*~ 4.408e+06

Do not underestimate the power of AM, it can be very
satisfying in a 100 seconds live loop like this one.

expr~ sin($v1/32)*($v1%5510<2000)*(sin($v1/1000)*0.5+0.5)*0.1

/1000 is the speed, *0.5+0.5 normalizes the sine to 0-1

Yes... we will need more and more space.

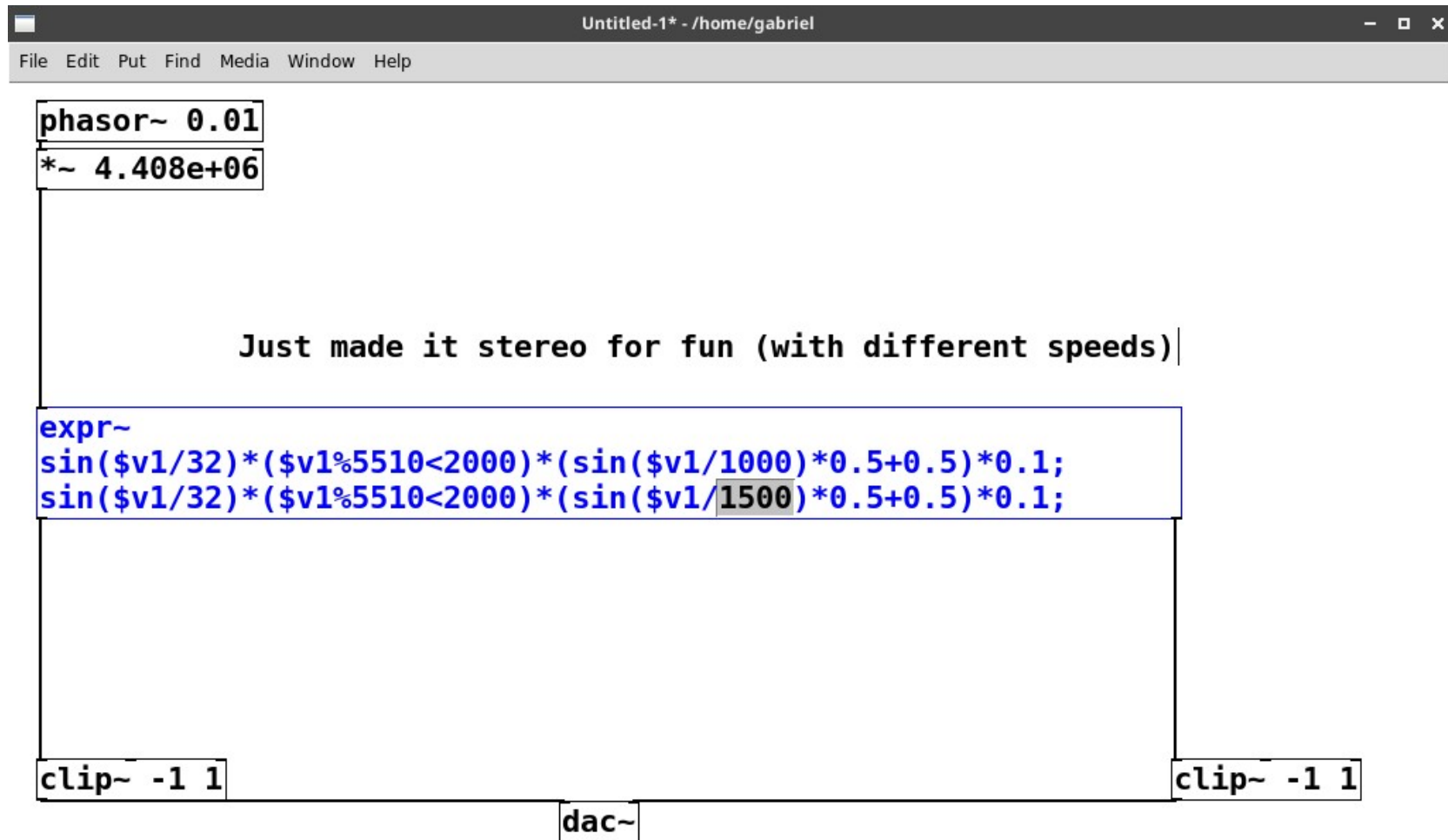
clip~ -1 1
dac~
  
```

De más está decir que podemos usarlo para controlar la amplitud sin disminuirla completamente y lograr un efecto como de **distintas intensidades o matices**. Por ejemplo:

```
(sin($v1/2000) * 0.7+0.3)
```

## Stereo

Aprovechá el stereo (o la cuadrafonía si la tenés) al máximo. Podés copiar los mismos sonidos con diferentes modulaciones y obtener resultados espaciales muy copados.

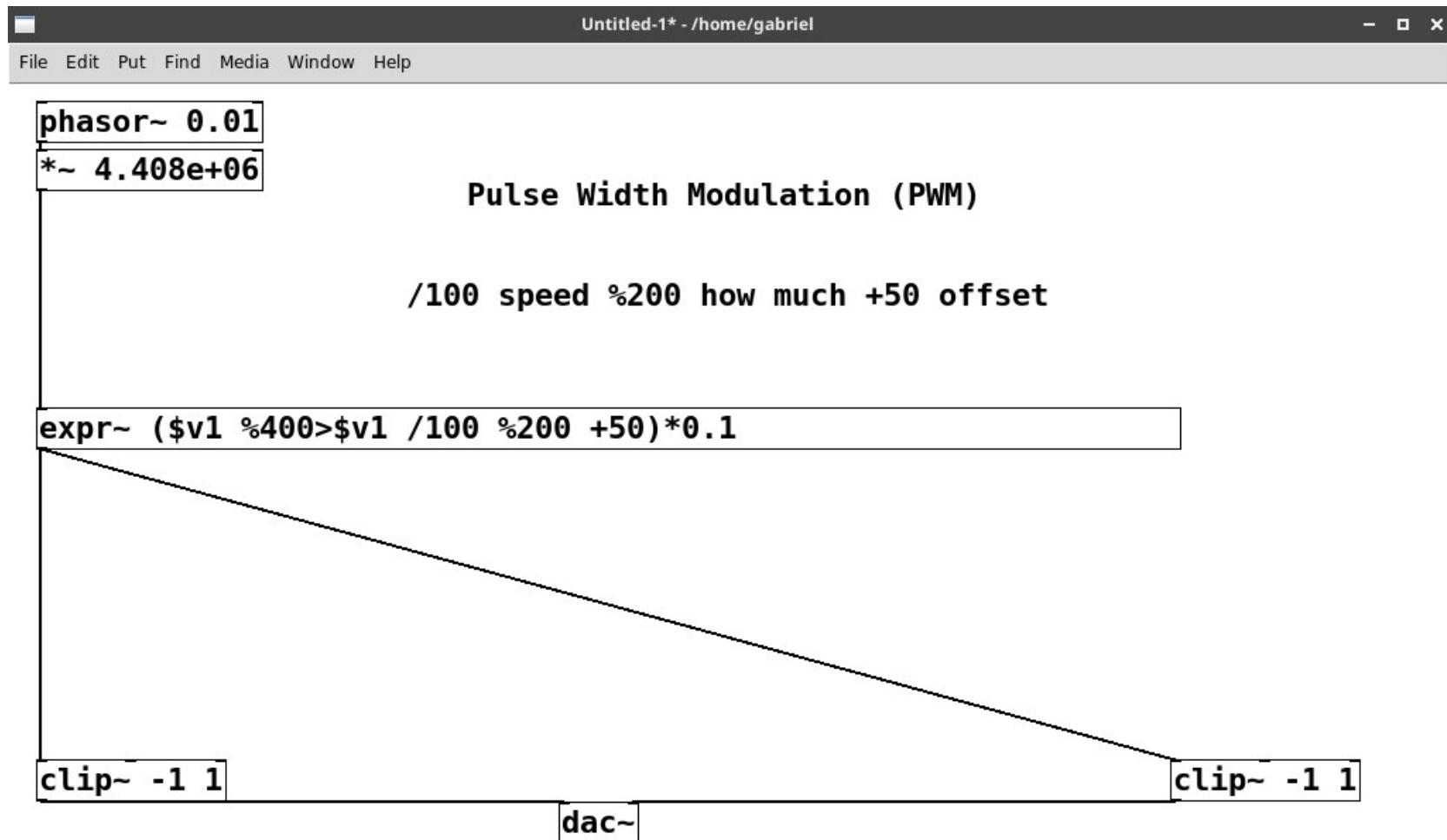


## PWM

```
($v1 %400>$v1 /100 %200 +50)
```

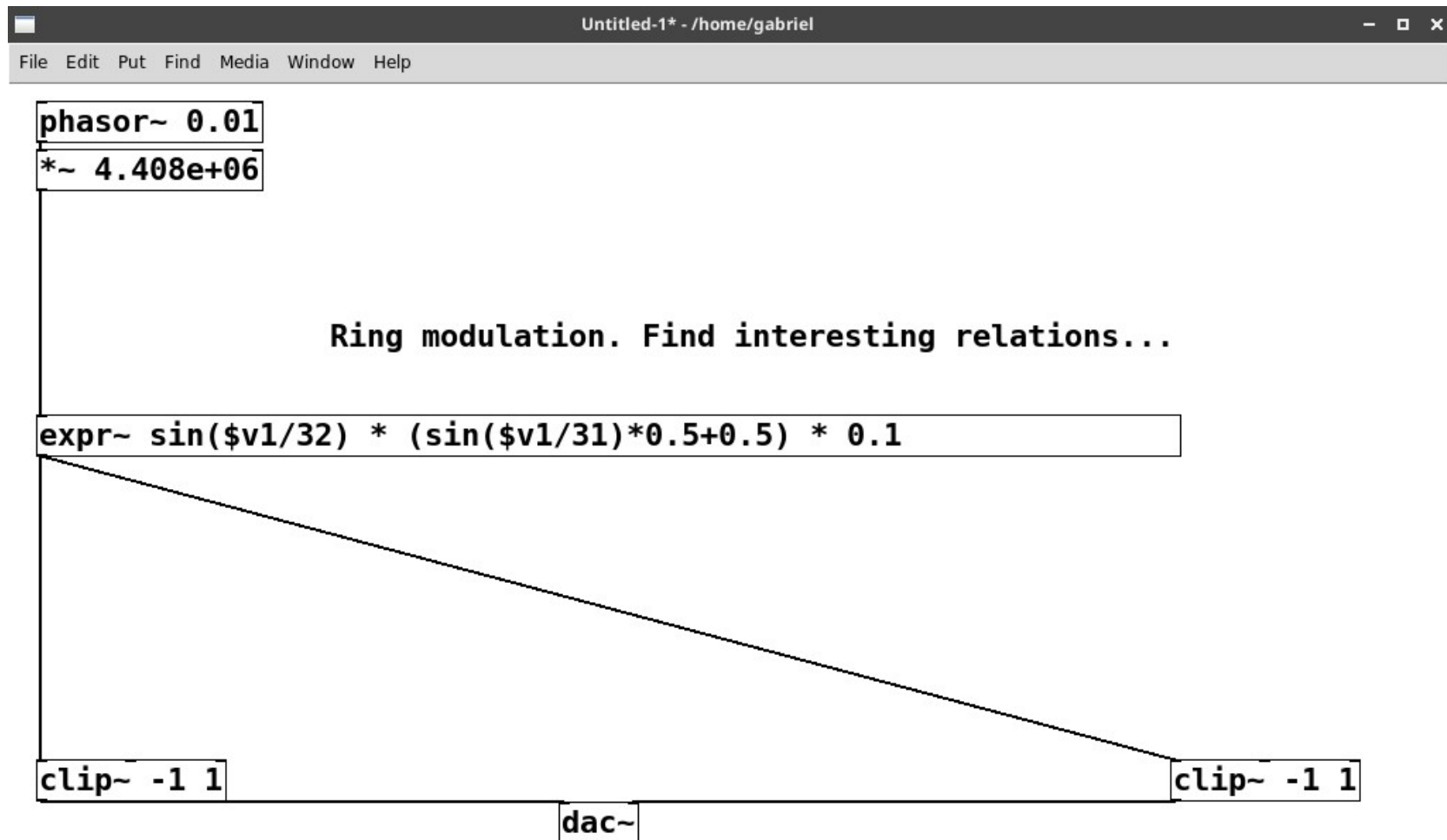
Modulando el ancho de pulso de una onda cuadrada conseguís una barrida de armónicos, desde timbres de chiptune, vibratos, desafinadas e incluso patrones rítmicos notables!





## Ring

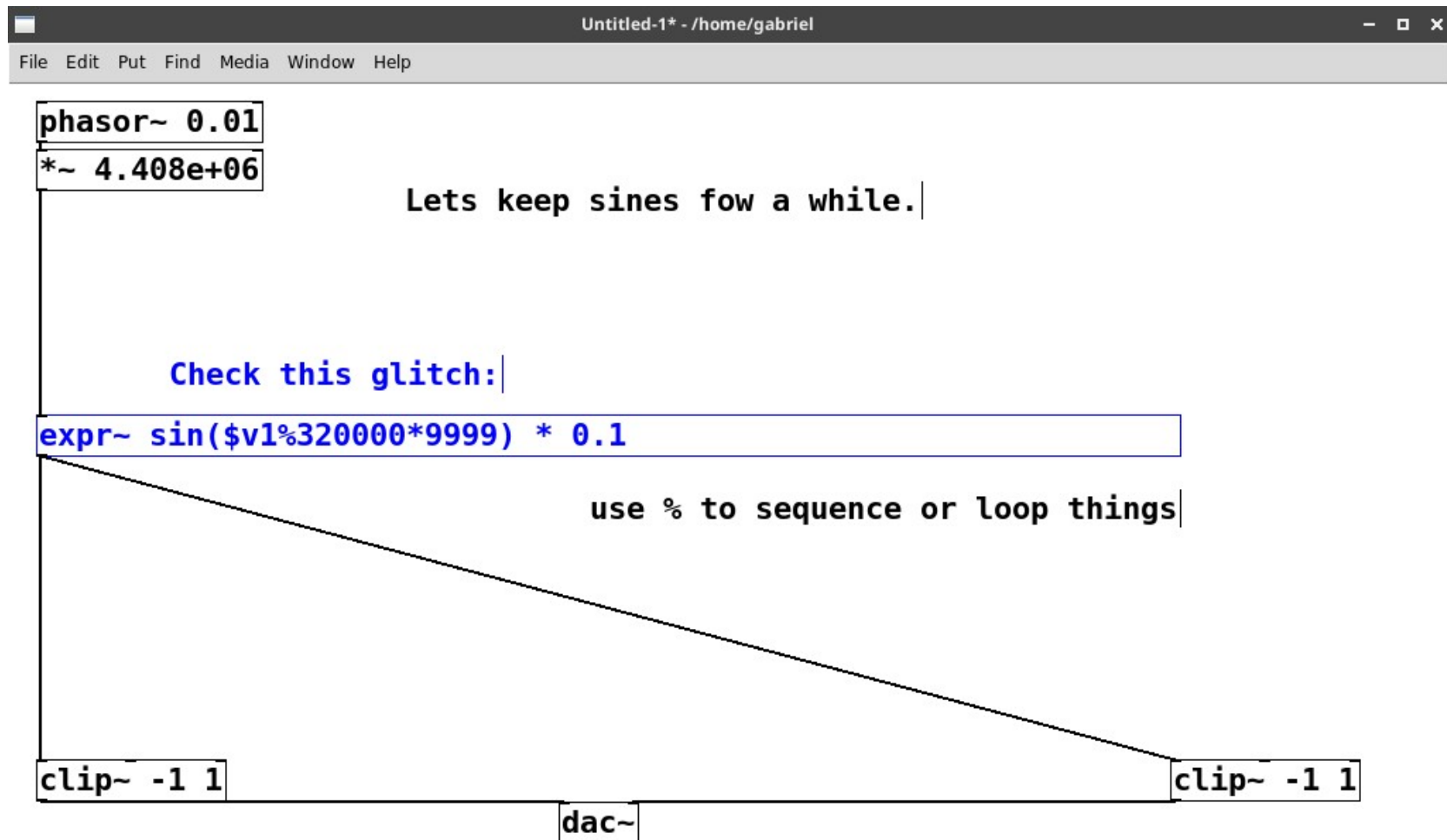
Es como una AM pero a frecuencias audibles, sobre todo aquellas que guarden una particular relación con la modulada. También logra **vibratos**, **detunings**, **tremolos**, etc.



## Glitching loops

```
sin($v1%320000 * 9999)
```

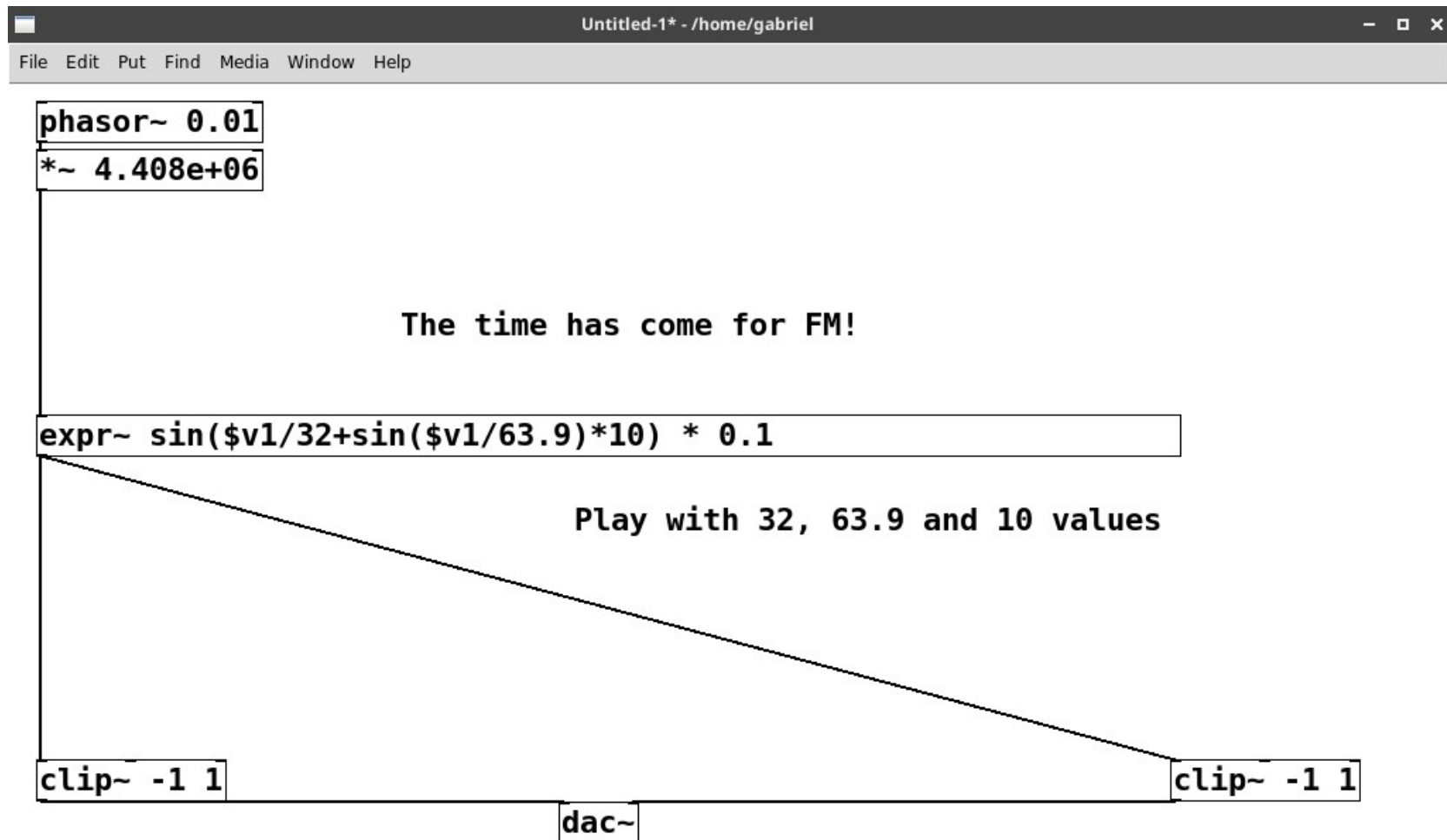
Creo que es mejor escucharlos que explicarlos, tampoco los entiendo muy bien, creo que están relacionados con nuestra especie de samplerate extraño.



## FM 2 operadores

¿Te acordás que te dije que no había que subestimar a las ondas sinusoidales? Bueno, cuando empezás a modularlas con otras ondas sinusoidales obtenés los resultados más ricos y complejos, siempre **siendo posible seguir modulando a niveles cada vez más locos**.

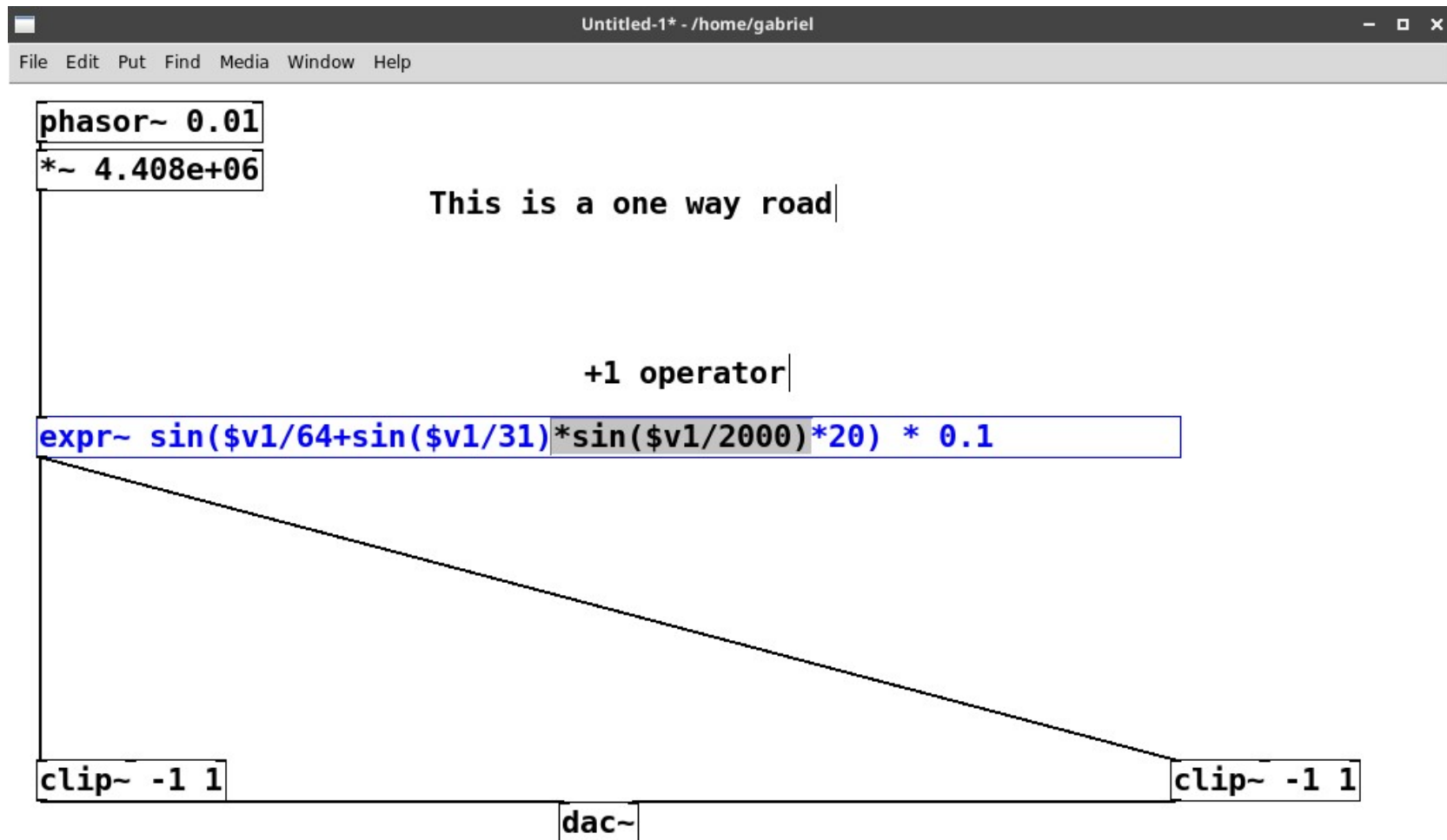
```
sin($v1/64+sin($v1/128) * 10)
```



Probá también modular, por ejemplo, la amplitud con una frecuencia a su vez modulada!

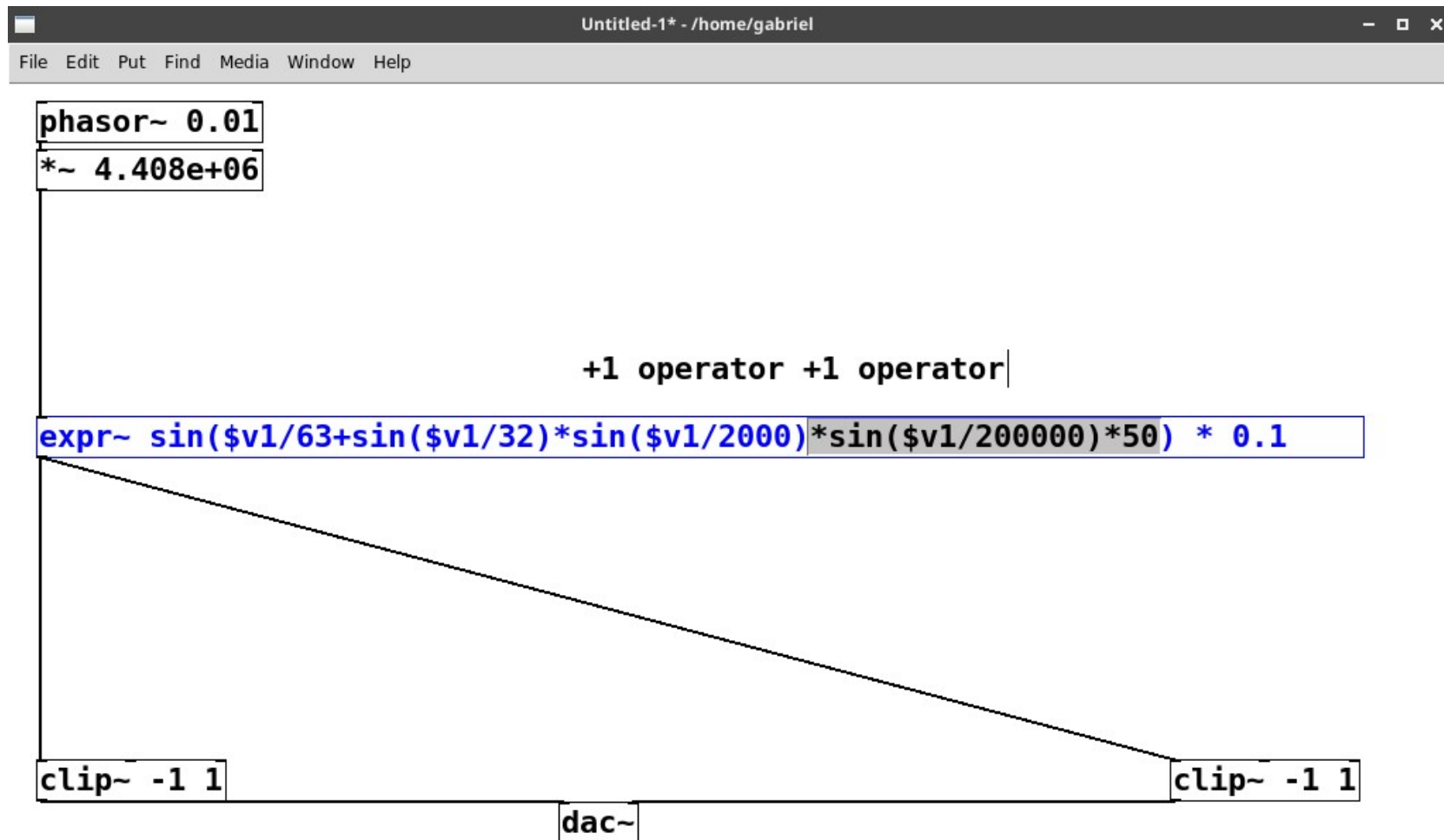
### FM 3 operadores

```
sin($v1/64+sin($v1/32) * sin($v1/2000) * 20)
```



## FM 4 operadores

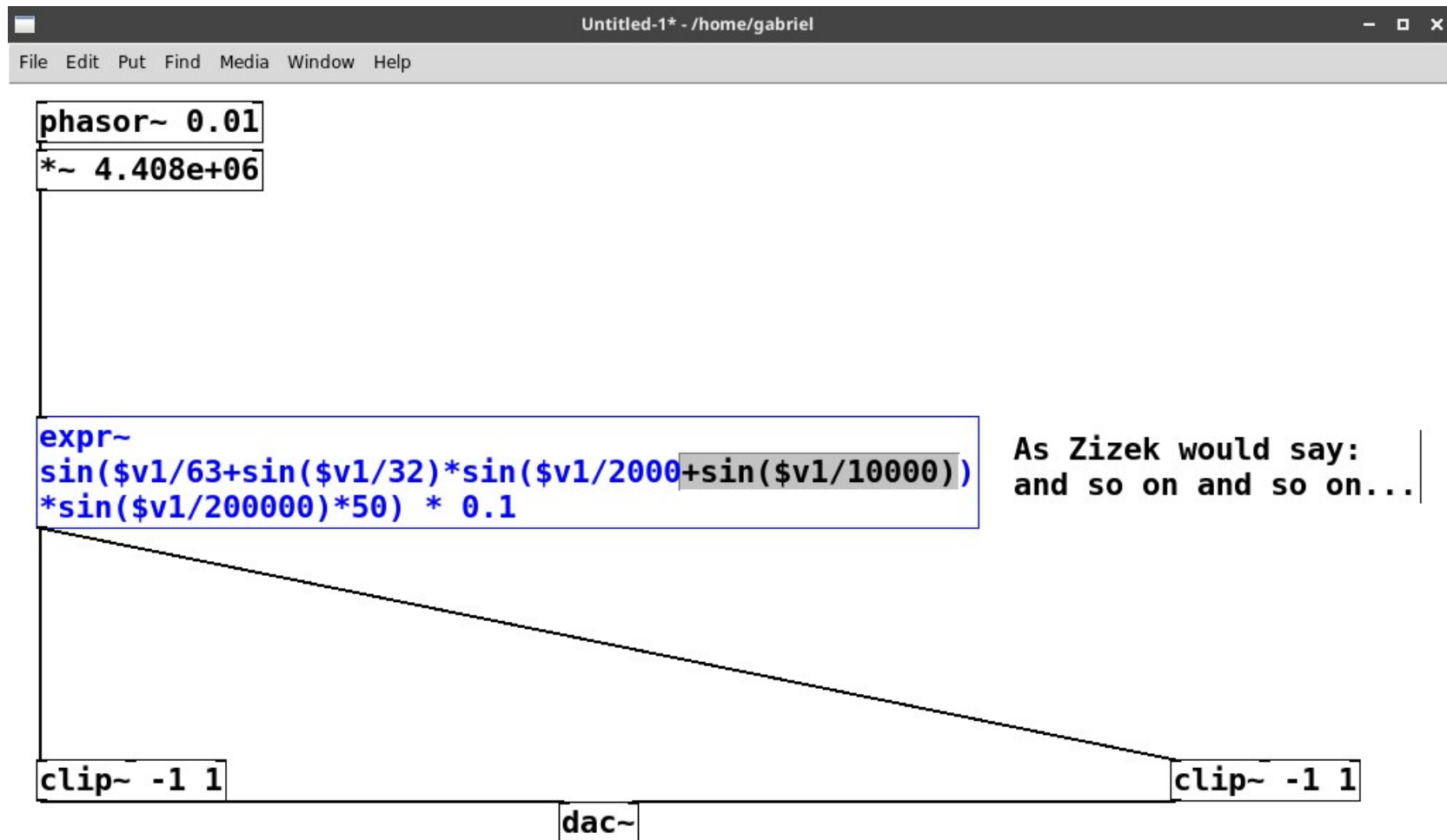
```
sin($v1/64+sin($v1/32) * sin($v1/2000) * sin($v1/200000) * 50)
```



## FM 5 operadores

Te advertí que no tiene fin...

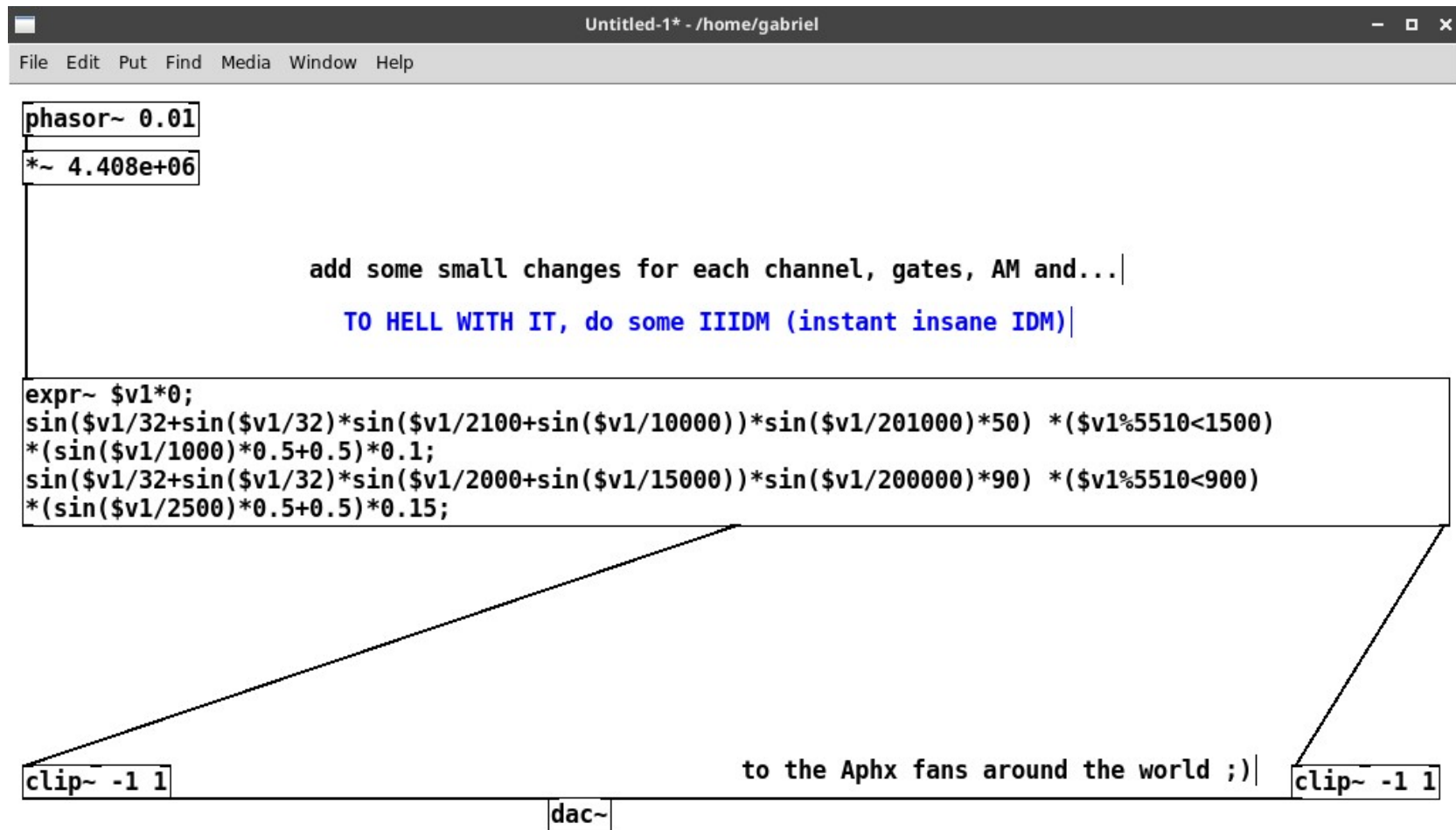
```
sin($v1/64+sin($v1/32) * sin($v1/2000+sin($v1/10000)) * sin($v1/200000) * 50) * 0.1
```



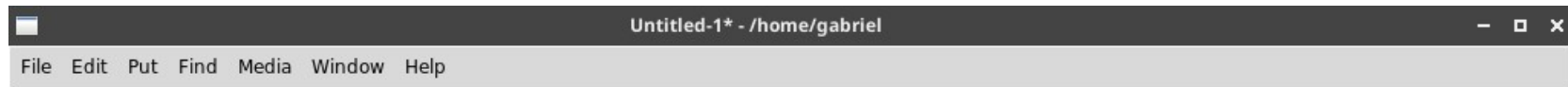
## Instant Insane IDM (Interludio)

*Un pequeño excursio musical...*

```
sin($v1/32+sin($v1/32) * sin($v1/2100+sin($v1/10000)) * sin($v1/201000) * 50) * (sin($v1/1000) * 0.5+0.5) *
($v1%5510<1500) * 0.1; sin($v1/32+sin($v1/32) * sin($v1/2000+sin($v1/15000)) * sin($v1/200000) * 50) * (sin($v1/2500) *
0.5+0.5) * ($v1%5510<900) * 0.1
```







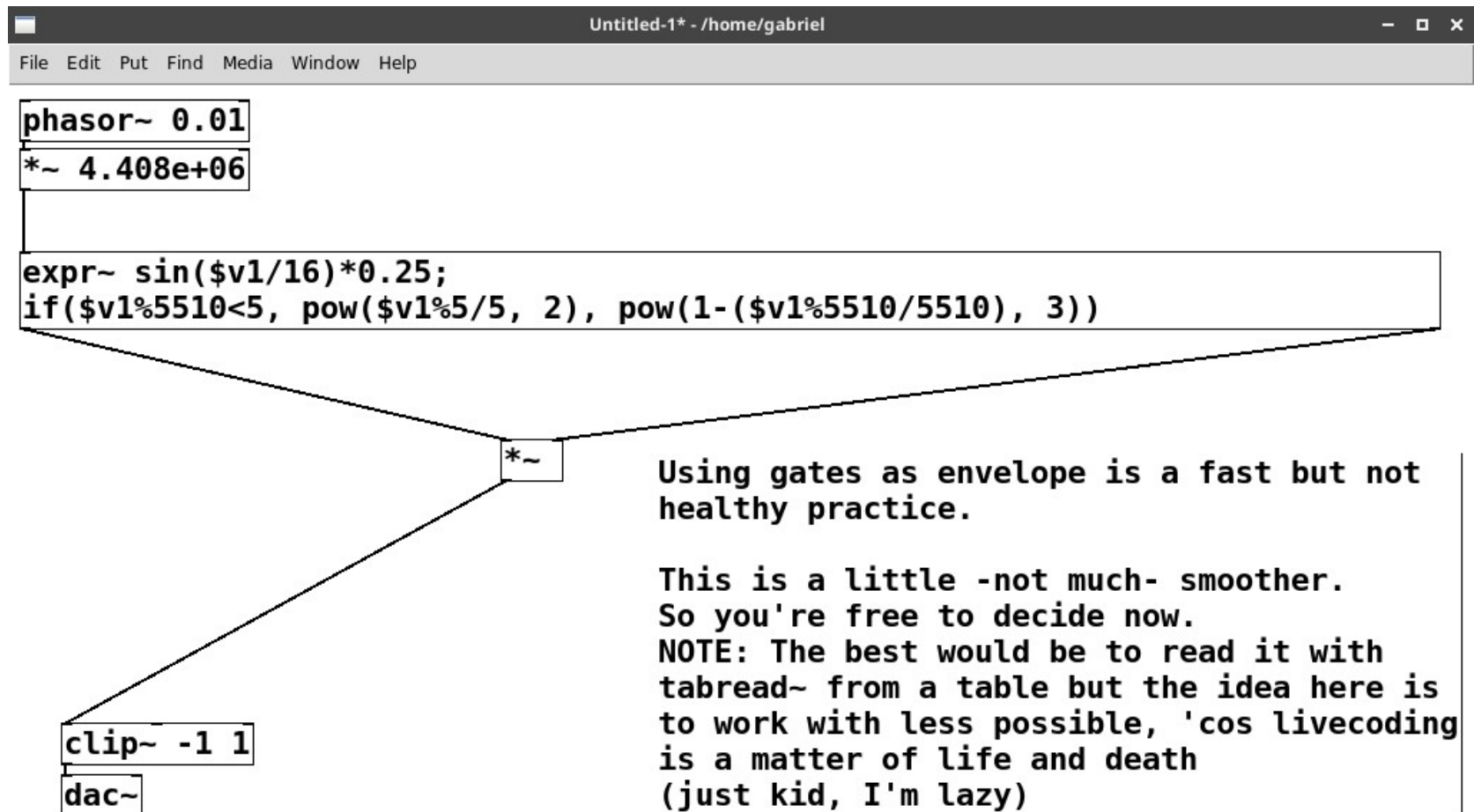
OK, sorry for that.  
Anyway you can always copy-paste  
from cypaste.txt

Now let's continue with envelopes  
and if your fingers are still  
alive I have some tips 4 you.

## Envolventes

En términos de amplitud es aconsejable trabajar con elevaciones *pow()*. La importancia de los envolventes en este caso particular es que si no suavizamos un poco los ataques y relajamientos generamos un montón de pops, clicks y cosas que explotan de armónicos y daños colaterales. Esta es una posible solución (para nada perfecta igual, es un tema que me preocupa un poco):

```
if($v1%5510<5, pow($v1%5/5, 2), pow(1-($v1%5510/5510), 3))
```



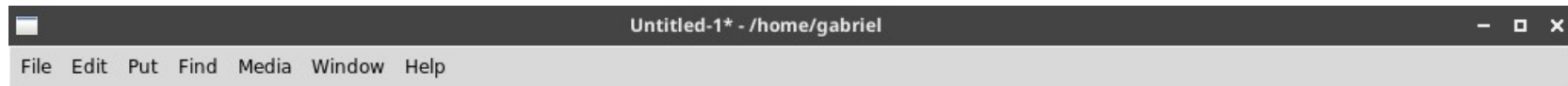
## Tips

Algunos tips finales de arpeggios y glitches que te pueden interesar:

```
($v1 / 5510 % 8 * 8)
```

```
sin($v1%150)
```

```
sin(8 * $v1%10)
```



Use it as an aggressive nonetheless worst kick in the world

```
expr~ ($v1%(5510*4)<150)*0.2
```

Make sequences with (\$v1 /5510 %X \*X) style formula like:

```
expr~ ($v1% ($v1 /5510 %32%7 * 100) < 80) * 0.05
```

```
expr~ sin($v1/($v1 /5510 %5 *8))*0.05
```

Manage gates with (5510\*X) and use logic

```
expr~ ($v1%(5510*8)>(5510*4)&&$v1%(5510*4)<500)
```

More glitchy stuff?=\$/IU\$/Ñ&&[]

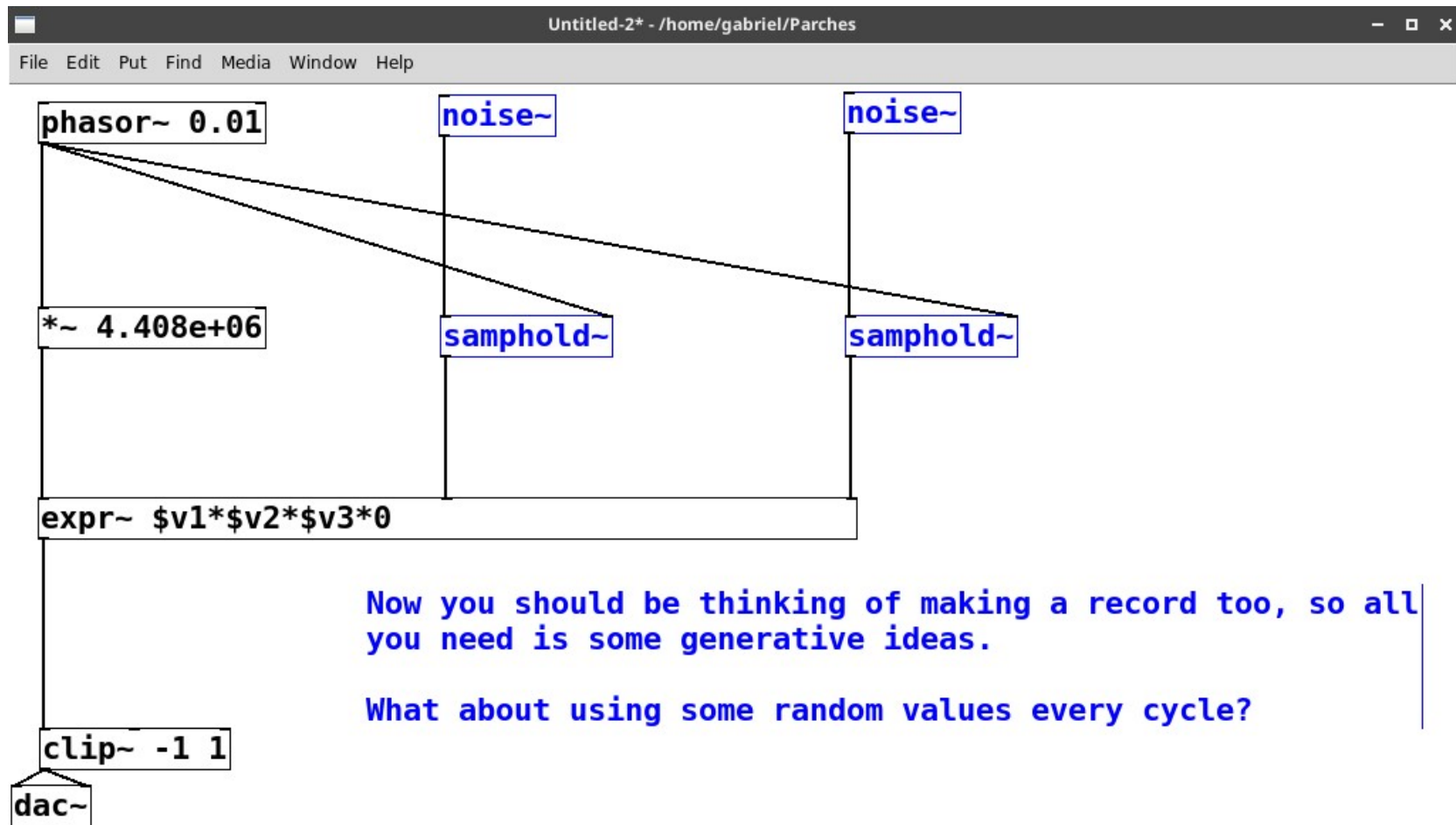
```
expr~ sin ($v1 %150) * 0.05
```

```
expr~ sin (8 * $v1 %10 +1) * 0.05
```

Almost done...

## Generatividad

Aunque salgamos de la consigna inicial, incluso haciendo livecoding, nada nos impide utilizar señales aleatorias para jugar un poco más (en este caso usamos *ruido* -1 1, lo normalizamos a 0 1 y lo sostenemos en sincro con el loop con *samphold~*):



Bye :)

Espero que te haya copado el tutorial y, sobre todo, que haya resultado inspirador. Te aliento a experimentar por tu cuenta y escribirme. Hay otros videos relacionados en mi canal de YouTube, chusmealo. Chau :)



**Well, it's about time to leave you.**

**Spread your pd wings and fly away!**

**bye~~~~~**